

PROGRAMIRANJE 2

**Milena Vujošević Janićić, Jelena Graovac,
Nina Radojičić, Ana Spasić,
Mirko Spasić, Anđelka Zečević**

PROGRAMIRANJE 2
Zbirka zadataka sa rešenjima

**Beograd
2016.**

Autori:

dr Milena Vujošević Janičić, docent na Matematičkom fakultetu u Beogradu

dr Jelena Graovac, docent na Matematičkom fakultetu u Beogradu

Nina Radojičić, asistent na Matematičkom fakultetu u Beogradu

Ana Spasić, asistent na Matematičkom fakultetu u Beogradu

Mirko Spasić, asistent na Matematičkom fakultetu u Beogradu

Andelka Zečević, asistent na Matematičkom fakultetu u Beogradu

PROGRAMIRANJE 2

Zbirka zadataka sa rešenjima

Izdavač: Matematički fakultet Univerziteta u Beogradu. Studentski trg 16, Beograd.

Za izdavača: *prof. dr Zoran Rakić*, dekan

Recenzenti:

dr Gordana Pavlović-Lažetić, redovni profesor na Matematičkom fakultetu u Beogradu

dr Dragan Urošević, naučni savetnik na Matematičkom institutu SANU

Obrada teksta, crteži i korice: *autori*.

Štampa: Copy Centar, Beograd. Tiraž 200.

СIP Каталогизација у публикацији

Народна библиотека Србије, Београд

004.4(075.8)(076)

004.432.2C(075.8)(076)

PROGRAMIRANJE 2 : zbirka zadataka sa rešenjima / Milena Vujošević

Jančić ... [et al.]. - Beograd : Matematički fakultet, 2016

(Beograd : Copy Centar). - VII, 361 str. ; 24 cm

Tiraž 200.

ISBN 978-86-7589-107-9

1. Вујошевић Јаничић, Милена 1980- [аутор]

а) Програмирање - Задаци б) Програмски језик "C"- Задаци

COBISS.SR-ID 221508876

©2016. Milena Vujošević Jančić, Jelena Graovac, Nina Radojičić, Ana Spasić, Mirko Spasić, Andelka Zečević

Ovo delo zaštićeno je licencom Creative Commons CC BY-NC-ND 4.0 (Attribution-NonCommercial-NoDerivatives 4.0 International License). Detalji licence mogu se videti na veb-adresi <http://creativecommons.org/licenses/by-nc-nd/4.0/>. Dozvoljeno je umnožavanje, distribucija i javno saopštavanje dela, pod uslovom da se navedu imena autora. Upotreba dela u komercijalne svrhe nije dozvoljena. Prerada, preoblikovanje i upotreba dela u sklopu nekog drugog nije dozvoljena.



Sadržaj

1	Algoritmi pretrage i sortiranja	ix
1.1	Algoritmi pretrage	ix
1.2	Algoritmi sortiranja	xiv
1.3	Bibliotečke funkcije pretrage i sortiranja	xxiv
1.4	Rešenja	xxviii

Predgovor

U okviru kursa *Programiranje 2* na Matematičkom fakultetu vežbaju se zadaci koji imaju za cilj da studente nauče rekurzivnom pristupu rešavanja problema, ispravnom radu sa pokazivačima i dinamički alociranom memorijom, osnovnim algoritmima pretraživanja i sortiranja, kao i radu sa dinamičkim strukturama podataka, poput listi i stabala. Zadaci koji se nalaze u ovoj zbirci predstavljaju objedinjen skup zadataka sa vežbi i praktikuma ovog kursa, kao i primere zadataka sa održanih ispita. Elektronska verzija zbirke i propratna rešenja u elektronskom formatu, dostupna su besplatno u okviru strane kursa www.programiranje2.matf.bg.ac.rs u skladu sa navedenom licencom.

U prvom poglavlju zbirke obrađene su uvodne teme koje obuhvataju osnovne tehnike koje se koriste u rešavanju svih ostalih zadataka u zbirci: podela koda po datotekama i rekurzivni pristup rešavanju problema. Takođe, u okviru ovog poglavlja dati su i osnovni algoritmi za rad sa bitovima. Drugo poglavlje je posvećeno pokazivačima: pokazivačkoj aritmetici, višedimenzionim nizovima, dinamičkoj alokaciji memorije i radu sa pokazivačima na funkcije. Treće poglavlje obrađuje algoritme pretrage i sortiranja, a četvrto dinamičke strukture podataka: liste i stabla. Dodatak sadrži najvažnije ispitne rokove iz jedne akademske godine. Većina zadataka je rešena, a teži zadaci su obeleženi zvezdicom.

Autori velikog broja zadataka ove zbirke su ujedno i autori same zbirke, ali postoje i zadaci za koje se ne može tačno utvrditi ko je originalni autor jer su zadacima davali svoje doprinose različiti asistenti koji su držali vežbe iz ovog kursa u prethodnih desetak godina. Zbog toga smatramo da je naš osnovni doprinos što smo objedinili, precizno formulisali, rešili i detaljno iskomentarisali sve najvažnije zadatke koji su potrebni za uspešno savlađivanje koncepata koji se obrađuju u okviru kursa. Takođe, formulacije zadataka smo obogatili primerima koji upotpunjuju razumevanje zahteva zadataka i koji omogućavaju čitaocu zbirke da proveriti sopstvena rešenja. Primeri su dati u obliku testova i interakcija sa programom. Testovi su svedene prirode i obuhvataju samo jednostavne ulaze i izlaze iz programa. Interakcija sa programom obuhvata naizmeničnu interakciju čovek-računar u kojoj su ulazi i izlazi isprepleteni. U zadacima koji zahtevaju

rad sa argumentima komandne linije, navedeni su i primeri poziva programa, a u zadacima koji demonstriraju rad sa datotekama, i primeri ulaznih ili izlaznih datoteka. Test primeri koji su navedeni uz ispitne zadatke u dodatku su oni koji su korišćeni za početno testiranje (koje prethodi ocenjivanju) studentskih radova na ispitima.

Neizmerno zahvaljujemo recenzentima, Gordani Pavlović Lažetić i Draganu Uroševiću, na veoma pažljivom čitanju rukopisa i na brojnim korisnim sugestijama. Takođe, zahvaljujemo studentima koji su svojim aktivnim učešćem u nastavi pomogli i doprineli uobličavanju ovog materijala.

Svi komentari i sugestije na sadržaj zbirke su dobrodošli i osećajte se slobodnim da ih pošaljete elektronskom poštom bilo kome od autora¹.

Autori

¹Adrese autora su: milena, jgraovac, nina, aspasic, mirko, andjelkaz, sa nastavkom @matf.bg.ac.rs

1

Algoritmi pretrage i sortiranja

1.1 Algoritmi pretrage

Zadatak 1.1 Napisati iterativne funkcije za pretragu nizova. Svaka funkcija treba da vrati indeks pozicije na kojoj je pronađen traženi broj ili vrednost -1 ukoliko broj nije pronađen.

- (a) Napisati funkciju `linarna_pretraga` koja vrši linearnu pretragu niza celih brojeva `a`, dužine `n`, tražeći u njemu broj `x`.
- (b) Napisati funkciju `binarna_pretraga` koja vrši binarnu pretragu sortiranog niza `a`, dužine `n`, tražeći u njemu broj `x`.
- (c) Napisati funkciju `interpolaciona_pretraga` koja vrši interpolacionu pretragu sortiranog niza `a`, dužine `n`, tražeći u njemu broj `x`.

Napisati i program koji generiše rastući niz slučajnih brojeva dimenzije n i pozivajući napisane funkcije traži broj x . Programu se kao prvi argument komandne linije prosleđuje prirodan broj n koji nije veći od 1000000 i broj x kao drugi argument komandne linije. Potrebna vremena za izvršavanje ovih funkcija dopisati u datoteku `vremena.txt`.

Test 1

POKRETANJE: ./a.out 1000000 23542	VREMENA.TXT
IZLAZ:	Dimenzija niza: 1000000
Linearna pretraga:	Linearna: 3615091 ns
Element nije u nizu	Binarna: 1536 ns
Binarna pretraga:	Interpolaciona: 558 ns
Element nije u nizu	
Interpolaciona pretraga:	
Element nije u nizu	

Test 2

POKRETANJE: ./a.out 100000 37842	VREMENA.TXT
IZLAZ:	Dimenzija niza: 1000000
Linearna pretraga:	Linearna: 3615091 ns
Element nije u nizu	Binarna: 1536 ns
Binarna pretraga:	Interpolaciona: 558 ns
Element nije u nizu	
Interpolaciona pretraga:	Dimenzija niza: 100000
Element nije u nizu	Linearna: 360803 ns
	Binarna: 1187 ns
	Interpolaciona: 628 ns

Zadatak 1.2 Napisati rekurzivne funkcije koje implementiraju algoritme linearne, binarne i interpolacione pretrage i program koji ih testira za brojeve koji se unose sa standardnog ulaza. Linearnu pretragu implementirati na dva načina, svođenjem pretrage na prefiks i na sufiks niza. Prvo se unosi broj koji se traži, a zatim sortirani elementi niza sve do kraja ulaza. Pretpostaviti da niz brojeva koji se unosi neće biti duži od 1024 elemenata.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
Unesite trazeni broj: 11
Unesite sortiran niz elemenata:
2 5 6 8 10 11 23
Linearna pretraga
Pozicija elementa je 5.
Binarna pretraga
Pozicija elementa je 5.
Interpolaciona pretraga
Pozicija elementa je 5.
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
Unesite trazeni broj: 14
Unesite sortiran niz elemenata:
10 32 35 43 66 89 100
Linearna pretraga
Element se ne nalazi u nizu.
Binarna pretraga
Element se ne nalazi u nizu.
Interpolaciona pretraga
Element se ne nalazi u nizu.
```

Zadatak 1.3 Napisati program koji preko argumenta komandne linije dobija ime datoteke koja sadrži sortirani spisak studenta po broju indeksa rastuće. Za svakog studenta u jednom redu stoje informacije o indeksu, imenu i prezimenu. Program učitava spisak studenata u niz i traži od korisnika indeks ili prezime studenta čije informacije se potom prikazuju na standardni izlaz. U slučaju više

studenata sa istim prezimenom prikazati informacije o prvom takvom. Odabir kriterijuma pretrage se vrši kroz poslednji argument komandne linije, koji može biti **-indeks** ili **-prezime**. U slučaju neuspešnih pretragi, štampati odgovarajuću poruku. Pretrage implementirati u vidu iterativnih funkcija što manje složeno-sti. Pretpostaviti da u datoteci neće biti više od 128 studenata i da su imena i prezimena svih kraća od 16 slova.

Primer 1

```
POKRETANJE: ./a.out datoteka.txt -indeks
DATOTEKA.TXT
20140003 Marina Petrovic
20140012 Stefan Mitrovic
20140032 Dejan Popovic
20140049 Mirko Brankovic
20140076 Sonja Stevanovic
20140104 Ivan Popovic
20140187 Vlada Stankovic
20140234 Darko Brankovic
```

```
INTERAKCIJA SA PROGRAMOM:
Unesite indeks studenta
cije informacije zelite:
20140076
Indeks: 20140076,
Ime i prezime: Sonja Stevanovic
```

Primer 2

```
POKRETANJE: ./a.out datoteka.txt -prezime
DATOTEKA.TXT
20140003 Marina Petrovic
20140012 Stefan Mitrovic
20140032 Dejan Popovic
20140049 Mirko Brankovic
20140076 Sonja Stevanovic
20140104 Ivan Popovic
20140187 Vlada Stankovic
20140234 Darko Brankovic
```

```
INTERAKCIJA SA PROGRAMOM:
Unesite prezime studenta
cije informacije zelite:
Popovic
Indeks: 20140032,
Ime i prezime: Dejan Popovic
```

Zadatak 1.4 Modifikovati zadatak 1.3 tako da tražene funkcije budu rekurzivne.

Zadatak 1.5 U datoteci koja se zadaje kao prvi argument komandne linije, nalaze se koordinate tačaka. U zavisnosti od prisustva opcija komandne linije (**-x** ili **-y**), pronaći onu koja je najbliža x , ili y osi, ili koordinatnom početku, ako nije prisutna nijedna opcija. Pretpostaviti da je broj tačaka u datoteci veći od 0 i ne veći od 1024.

Test 1

```
POKRETANJE: ./a.out dat.txt -x

DAT.TXT
12 53
2.342 34.1
-0.3 23
-1 23.1
123.5 756.12

IZLAZ:
-0.3 23
```

Test 2

```
POKRETANJE: ./a.out dat.txt

DAT.TXT
12 53
2.342 34.1
-0.3 23
-1 2.1
123.5 756.12

IZLAZ:
-1 2.1
```

Zadatak 1.6 Napisati funkciju koja određuje nulu funkcije $\cos(x)$ na intervalu $[0, 2]$ metodom polovljenja intervala. Algoritam se završava kada se vrednost kosinusne funkcije razlikuje za najviše 0.001 od nule. UPUTSTVO: *Koristiti metod polovljenja intervala (algoritam analogan algoritmu binarne pretrage).* NAPOMENA: *Ovaj metod se može primeniti na funkciju $\cos(x)$ na intervalu $[0, 2]$ zato što je ona na ovom intervalu neprekidna, i vrednosti funkcije na krajevima intervala su različitog znaka.*

Test 1

```
IZLAZ:
1.57031
```

Zadatak 1.7 Napisati funkciju koja metodom polovljenja intervala određuje nulu izabrane funkcije na proizvoljnom intervalu sa tačnošću *epsilon*. Ime funkcije se zadaje kao prvi argument komandne linije, a interval i tačnost se unose sa standardnog ulaza. Pretpostaviti da je izabrana funkcija na tom intervalu neprekidna. UPUTSTVO: *U okviru algoritma pretrage koristiti pokazivač na odgovarajuću funkciju (na primer, kao u zadatku ??).*

Primer 1

```
POKRETANJE: ./a.out cos

INTERAKCIJA SA PROGRAMOM:
Unesite krajeve intervala: 0 2
Unesite preciznost: 0.001
1.57031
```

Primer 2

```
POKRETANJE: ./a.out sin

INTERAKCIJA SA PROGRAMOM:
Unesite krajeve intervala: 1 5
Unesite preciznost: 0.00001
3.1416
```

Primer 3

```
POKRETANJE: ./a.out tan

INTERAKCIJA SA PROGRAMOM:
Unesite krajeve intervala: -1.1 1
Unesite preciznost: 0.00001
3.8147e-06
```

Primer 4

```
POKRETANJE: ./a.out sin

INTERAKCIJA SA PROGRAMOM:
Unesite krajeve intervala: 1 3
Funkcija sin na intervalu [1, 3]
ne zadovoljava uslove
```

Zadatak 1.8 Napisati funkciju koja u rastuće sortiranom nizu celih brojeva binarnom pretragom pronalazi indeks prvog elementa većeg od nule. Ukoliko nema elemenata većih od nule, funkcija kao rezultat vraća -1 . Napisati program koji testira ovu funkciju za rastući niz celih brojeva koji se učitavaju sa standardnog ulaza. Niz neće biti duži od 256, i njegovi elementi se unose sve do kraja ulaza.

<i>Test 1</i>	<i>Test 2</i>	<i>Test 3</i>
<pre> ULAZ: -151 -44 5 12 13 15 IZLAZ: 2 </pre>	<pre> ULAZ: -100 -15 -11 -8 -7 -5 IZLAZ: -1 </pre>	<pre> ULAZ: -100 -15 0 13 55 124 258 315 516 7000 IZLAZ: 3 </pre>

Zadatak 1.9 Napisati funkciju koja u opadajuće sortiranom nizu celih brojeva binarnom pretragom pronalazi indeks prvog elementa manjeg od nule. Ukoliko nema elemenata manjih od nule, funkcija kao rezultat vraća -1 . Napisati program koji testira ovu funkciju za opadajući niz celih brojeva koji se učitavaju sa standardnog ulaza. Niz neće biti duži od 256, i njegovi elementi se unose sve do kraja ulaza.

<i>Test 1</i>	<i>Test 2</i>	<i>Test 3</i>
<pre> ULAZ: 151 44 5 -12 -13 -15 IZLAZ: 3 </pre>	<pre> ULAZ: 100 55 15 0 -15 -124 -155 -258 -315 -516 IZLAZ: 4 </pre>	<pre> ULAZ: 100 15 11 8 7 5 4 3 2 IZLAZ: -1 </pre>

Zadatak 1.10 Napisati funkciju koja određuje ceo deo logaritma za osnovu 2 datog neoznačenog celog broja koristeći samo bitske i relacione operatore.

- Napisati funkciju linearne složenosti koja određuje logaritam pomeranjem broja udesno.
- Napisati funkciju logaritmske složenosti koja određuje logaritam koristeći binarnu pretragu.

Napisati program koji testira napisane funkcije. Sa standardnog ulaza učitati pozitivan ceo broj, a na standardni izlaz ispisati njegov logaritam.

Test 1	Test 2	Test 3
<pre> ULAZ: 4 IZLAZ: 2 2 </pre>	<pre> ULAZ: 17 IZLAZ: 4 4 </pre>	<pre> ULAZ: 1031 IZLAZ: 10 10 </pre>

*** Zadatak 1.11** U prvom kvadrantu dato je $1 \leq N \leq 10000$ duži svojim koordinatama. Duži mogu da se seku, preklapaju, itd. Napisati program koji pronalazi najmanji ugao $0 \leq \alpha \leq 90^\circ$, na dve decimale, takav da je suma dužina duži sa obe strane polupoluprave iz koordinatnog početka pod uglom α jednak. Neke duži bivaju presečene, a neke ne. Program prvo učitava broj N , a zatim i same koordinate temena duži. UPUTSTVO: *Vršiti binarnu pretragu intervala* $[0, 90^\circ]$.

Primer 1	Primer 2	Primer 3
<pre> INTERAKCIJA SA PROGRAMOM: Unesite broj tacaka: 2 Unesite koordinate tacaka: 2 0 2 1 1 2 2 2 26.57 </pre>	<pre> INTERAKCIJA SA PROGRAMOM: Unesite broj tacaka: 2 Unesite koordinate tacaka: 1 0 1 1 0 1 1 1 45 </pre>	<pre> INTERAKCIJA SA PROGRAMOM: Unesite broj tacaka: 3 Unesite koordinate tacaka: 1 0 1 1 2 0 2 1 1 2 2 2 26.57 </pre>

1.2 Algoritmi sortiranja

Zadatak 1.12 Napraviti biblioteku koja implementira algoritme sortiranja nizova celih brojeva. Biblioteka treba da sadrži algoritam sortiranja izborom (engl. **selection sort**), sortiranja spajanjem (engl. **merge sort**), brzog sortiranja (engl. **quick sort**), mehurastog sortiranja (engl. **bubble sort**), sortiranja direktnim umetanjem (engl. **insertion sort**) i sortiranja umetanjem sa inkrementom (engl. **shell sort**). Upotrebiti biblioteku kako bi se napravilo poređenje efikasnosti različitih algoritama sortiranja. Efikasnost meriti na slučajno generisanim nizovima, na rastuće sortiranim nizovima i na opadajuće sortiranim nizovima. Izbor algoritma, veličine i početnog rasporeda elemenata niza birati kroz argumente komandne linije. Moguće opcije kojima se bira algoritam sortiranja su: **-m** za sortiranje spajanjem, **-q** za brzo sortiranje, **-b** za mehurasto, **-i** za sortiranje direktnim umetanjem ili **-s** za sortiranje umetanjem sa inkrementom. U slučaju da nije prisutna ni jedna od ovih opcija, niz sortirati algoritmom sortiranja izborom. Niz koji se sortira generisati neopadajuće ako je prisutna opcija

`-r`, nerastuće ako je prisutna opcija `-o` ili potpuno slučajno ako nema nijedne opcije. Vreme meriti programom `time`. Analizirati porast vremena sa porastom dimenzije n .

<p><i>Test 1</i></p> <pre> POKRETANJE: time ./a.out 200000 IZLAZ: real 0m42.168s user 0m42.100s sys 0m0.000s </pre>	<p><i>Test 2</i></p> <pre> POKRETANJE: time ./a.out 400000 IZLAZ: real 2m48.395s user 2m48.128s sys 0m0.000s </pre>	<p><i>Test 3</i></p> <pre> POKRETANJE: time ./a.out 800000 IZLAZ: real 11m13.703s user 11m12.636s sys 0m0.000s </pre>
<p><i>Test 4</i></p> <pre> POKRETANJE: time ./a.out 800000 -r IZLAZ: real 11m21.533s user 11m20.436s sys 0m0.020s </pre>	<p><i>Test 5</i></p> <pre> POKRETANJE: time ./a.out 800000 -q IZLAZ: real 0m0.159s user 0m0.156s sys 0m0.000s </pre>	<p><i>Test 6</i></p> <pre> POKRETANJE: time ./a.out 800000 -m IZLAZ: real 0m0.137s user 0m0.136s sys 0m0.000s </pre>

Zadatak 1.13 Dve niske su anagrami ako se sastoje od istog broja istih karaktera. Napisati program koji proverava da li su dve niske karaktera anagrami. Niske se zadaju sa standardnog ulaza i neće biti duže od 127 karaktera. UPUTSTVO: *Napisati funkciju koja sortira slova unutar niske karaktera, a zatim za sortirane niske proveriti da li su identične.*

<p><i>Primer 1</i></p> <pre> INTERAKCIJA SA PROGRAMOM: Unesite prvu nisku anagram Unesite drugu nisku ramgana jesu </pre>	<p><i>Primer 2</i></p> <pre> INTERAKCIJA SA PROGRAMOM: Unesite prvu nisku anagram Unesite drugu nisku anagrm nisu </pre>	<p><i>Primer 3</i></p> <pre> INTERAKCIJA SA PROGRAMOM: Unesite prvu nisku test Unesite drugu nisku tset jesu </pre>
--	---	--

Zadatak 1.14 U datom nizu brojeva treba pronaći dva broja koja su na najmanjem rastojanju. Niz se zadaje sa standardnog ulaza, sve do kraja ulaza, ali neće sadržati više od 256 i manje od 2 elemenata. Na izlaz ispisati razliku pronađena dva broja. UPUTSTVO: *Prvo sortirati niz.* NAPOMENA: *Koristiti biblioteku za sortiranje celih brojeva iz zadatka 1.12.*

<i>Test 1</i>	<i>Test 2</i>	<i>Test 3</i>
<pre> ULAZ: 23 64 123 76 22 7 IzLAZ: 1 </pre>	<pre> ULAZ: 21 654 65 123 65 12 61 IzLAZ: 0 </pre>	<pre> ULAZ: 34 30 IzLAZ: 4 </pre>

Zadatak 1.15 Napisati program koji pronalazi broj koji se najviše puta pojavljivao u datom nizu. Niz se zadaje sa standardnog ulaza sve do kraja ulaza i neće biti duži od 256 i kraći od jednog elemenata. UPUTSTVO: *Prvo sortirati niz, a zatim naći najdužu sekvencu jednakih elemenata.* NAPOMENA: *Koristiti biblioteku za sortiranje celih brojeva iz zadatka 1.12.*

<i>Test 1</i>	<i>Test 2</i>	<i>Test 3</i>
<pre> ULAZ: 4 23 5 2 4 6 7 34 6 4 5 IzLAZ: 4 </pre>	<pre> ULAZ: 2 4 6 2 6 7 99 1 IzLAZ: 2 </pre>	<pre> ULAZ: 123 IzLAZ: 123 </pre>

Zadatak 1.16 Napisati funkciju koja proverava da li u datom nizu postoje dva elementa čiji zbir je jednak zadatom celom broju. Napisati i program koji testira ovu funkciju. U programu se prvo učitava broj, a zatim i niz. Elementi niza se unose sve do kraja ulaza. Pretpostaviti da u niz neće biti uneto više od 256 brojeva. UPUTSTVO: *Prvo sortirati niz.* NAPOMENA: *Koristiti biblioteku za sortiranje celih brojeva iz zadatka 1.12.*

<i>Primer 1</i>	<i>Primer 2</i>	<i>Primer 3</i>
<pre> INTERAKCIJA SA PROGRAMOM: Unesite trazeni zbir: 34 Unesite elemente niza: 134 4 1 6 30 23 da </pre>	<pre> INTERAKCIJA SA PROGRAMOM: Unesite trazeni zbir: 12 Unesite elemente niza: 53 1 43 3 56 13 ne </pre>	<pre> INTERAKCIJA SA PROGRAMOM: Unesite trazeni zbir: 52 Unesite elemente niza: 52 ne </pre>

Zadatak 1.17 Napisati funkciju potpisa `int merge(int *niz1, int dim1, int *niz2, int dim2, int *niz3, int dim3)` koja prima dva sortirana niza, i na osnovu njih pravi novi sortirani niz koji sadrži elemente oba niza. Treća dimenzija predstavlja veličinu niza u koji se smešta rezultat. Ako je ona manja od potrebne dužine, funkcija vraća -1 kao indikator neuspeha, inače vraća 0. Napisati zatim program koji testira ovu funkciju. Nizovi se unose sa standardnog ulaza sve dok se ne unese 0. Može se pretpostaviti da će njihove dimenzije biti

manje od 256.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
Unesite elemente prvog niza:
3 6 7 11 14 35 0
Unesite elemente drugog niza:
3 5 8 0
3 3 5 6 7 8 11 14 35
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
Unesite elemente prvog niza:
1 4 7 0
Unesite elemente drugog niza:
9 11 23 54 75 0
1 4 7 9 11 23 54 75
```

Zadatak 1.18 Napisati program koji čita sadržaj dve datoteke od kojih svaka sadrži spisak imena i prezimena studenata iz jedne od dve grupe, rastuće sortiran po imenima, a u slučaju istog imena po prezimenima, i kreira jedinstven spisak studenata sortiranih takođe po istom kriterijumu. Program dobija nazive datoteka iz komandne linije i jedinstveni spisak upisuje u datoteku `ceo-tok.txt`. Pretpostaviti da ime studenta nije duže od 10, a prezime od 15 karaktera.

Test 1

```
POKRETANJE: ./a.out prvi-deo.txt drugi-deo.txt

PRVI-DEO.TXT
Andrija Petrovic
Anja Ilic
Ivana Markovic
Lazar Micic
Nenad Brankovic
Sofija Filipovic
Uros Milic
Vladimir Savic

DRUGI-DEO.TXT
Aleksandra Cvetic
Bojan Golubovic
Dragan Markovic
Filip Dukic
Ivana Stankovic
Marija Stankovic
Ognjen Peric

CEO-TOK.TXT
Aleksandra Cvetic
Andrija Petrovic
Anja Ilic
Bojan Golubovic
Dragan Markovic
Filip Dukic
Ivana Markovic
Ivana Stankovic
Lazar Micic
Marija Stankovic
Nenad Brankovic
Ognjen Peric
Sofija Filipovic
Vladimir Savic
Uros Milic
```

Zadatak 1.19 Napisati funkcije koje sortiraju niz struktura tačaka na osnovu sledećih kriterijuma: (i) njihovog rastojanja od koordinatnog početka, (ii) x koordinata tačaka, (iii) y koordinata tačaka. Napisati program koji učitava niz tačaka iz datoteke čije se ime zadaje kao drugi argument komandne linije, i u zavisnosti od prisutnih opcija (prvi argument) u komandnoj liniji (`-o`, `-x` ili `-y`) sortira tačke po jednom od prethodna tri kriterijuma i rezultat upisuje u datoteku čije se ime zadaje kao treći argument komandne linije. U ulaznoj datoteci nije zadato više od 128 tačaka.

Test 1

```
POKRETANJE: ./a.out -x in.txt out.txt
```

```
IN.TXT
```

```
3 4
11 6
7 3
2 82
-1 6
```

```
OUT.TXT
```

```
-1 6
2 82
3 4
7 3
11 6
```

Test 2

```
POKRETANJE: ./a.out -o in.txt out.txt
```

```
IN.TXT
```

```
3 4
11 6
7 3
2 82
-1 6
```

```
OUT.TXT
```

```
3 4
-1 6
7 3
11 6
2 82
```

Zadatak 1.20 Napisati program koji učitava imena i prezimena građana iz datoteke `biracki-spisak.txt` i kreira biračke spiskove. Jedan birački spisak je sortiran po imenu građana, a drugi po prezimenu. Program treba da ispisuje koliko građana ima isti redni broj u oba biračka spiska. Pretpostaviti da u će u datoteci biti najviše 1000 građana, da je za ime, odnosno prezime građana dovoljno 15 karaktera, da se nijedno ime i prezime ne pojavljuje više od jednom.

Test 1

```
BIRACKI-SPISAK.TXT
```

```
Bojan Golubovic
Andrija Petrovic
Anja Ilic
Aleksandra Cvetic
Dragan Markovic
Ivana Markovic
Lazar Micic
Marija Stankovic
Filip Dukic
```

```
IZLAZ:
```

```
3
```

Test 2

```
BIRACKI-SPISAK.TXT
```

```
Milan Milicevic
```

```
IZLAZ:
```

```
1
```

Test 3

```
DATOTEKA BIRACKI-SPISAK.TXT
```

```
NE POSTOJI
```

```
IZLAZ ZA GREŠKE:
```

```
Neuspesno otvaranje
datoteke za citanje
```

Zadatak 1.21 Definisati strukturu koja čuva imena, prezimena i godišta dece. Napisati funkciju koja sortira niz dece po godištu, a decu istog godišta sortira leksikografski po prezimenu i imenu. Napisati program koji učitava podatke o deci koji se nalaze u datoteci čije se ime zadaje kao prvi argument komandne linije, sortira ih i sortirani niz upisuje u datoteku čije se ime zadaje kao drugi argument komandne linije. Pretpostaviti da u ulaznoj datoteci nisu zadati podaci o više od 128 dece i da su imena i prezimena niske karaktera koje nisu duže od 30 karaktera.

Test 1

```
POKRETANJE: ./a.out in.txt out.txt
```

```
IN.OUT
```

```
Petar Petrovic 2007  
Milica Antonic 2008  
Ana Petrovic 2007  
Ivana Ivanovic 2009  
Dragana Markovic 2010  
Marija Antic 2007
```

```
OUT.TXT
```

```
Marija Antic 2007  
Ana Petrovic 2007  
Petar Petrovic 2007  
Milica Antonic 2008  
Ivana Ivanovic 2009  
Dragana Markovic 2010
```

Test 2

```
POKRETANJE: ./a.out in.txt out.txt
```

```
IN.OUT
```

```
Milijana Maric 2009
```

```
OUT.TXT
```

```
Milijana Maric 2009
```

Zadatak 1.22 Napisati funkciju koja sortira niz niski po broju suglasnika u niski. Ukoliko reči imaju isti broj suglasnika, sortirati ih po dužini niske rastuće, a ukoliko su i dužine jednake onda leksikografski rastuće. Napisati program koji testira ovu funkciju za niske koje se zadaju u datoteci `niske.txt`. Pretpostaviti da u nizu nema više od 128 elemenata, kao i da svaka niska sadrži najviše 31 karakter.

Test 1

```
NISKE.TXT
```

```
ana petar andjela milos nikola aleksandar ljubica matej milica
```

```
IZLAZ:
```

```
ana matej milos petar milica nikola andjela ljubica aleksandar
```

Zadatak 1.23 Napisati program koji simulira rad kase u prodavnici. Kupci prilaze kasi, a prodavac unošenjem bar-koda kupljenog proizvoda dodaje njegovu cenu na ukupan račun. Na kraju, program ispisuje ukupnu vrednost svih proizvoda. Sve artikle, zajedno sa bar-kodovima, proizvođačima i cenama učitati iz datoteke `artikli.txt`. Pretraživanje niza artikala vršiti binarnom pretragom.

Primer 1

```
ARTIKLI.TXT
1001 Keks Jaffa 120
2530 Napolitanke Bambi 230
0023 MedenoSrce Pionir 150
2145 Pardon Marbo 70

INTERAKCIJA SA PROGRAMOM:
Asortiman:
KOD Naziv artikla Ime proizvođača Cena
23 MedenoSrce Pionir 150.00
1001 Keks Jaffa 120.00
2145 Pardon Marbo 70.00
2530 Napolitanke Bambi 230.00
-----
- Za kraj za kraj rada kase, pritisnite CTRL+D!
- Za nov racun unesite kod artikla:

1001
Trazili ste: Keks Jaffa 120.00
Unesite kod artikla [ili 0 za prekid]: 23
Trazili ste: MedenoSrce Pionir 150.00
Unesite kod artikla [ili 0 za prekid]: 0

UKUPNO: 270.00 dinara.

-----
- Za kraj za kraj rada kase, pritisnite CTRL+D!
- Za nov racun unesite kod artikla:

232
Greska: Ne postoji proizvod sa traženim kodom!
Unesite kod artikla [ili 0 za prekid]: 2530
Trazili ste: Napolitanke Bambi 230.00
Unesite kod artikla [ili 0 za prekid]: 0

UKUPNO: 230.00 dinara.

-----
- Za kraj za kraj rada kase, pritisnite CTRL+D!
- Za nov racun unesite kod artikla:

Kraj rada kase!
```

Zadatak 1.24 Napisati program koji iz datoteke `aktivnost.txt` čita podatke o aktivnostima studenata na praktikumima i upisuje tri spiska redom u datoteke `dat1.txt`, `dat2.txt` i `dat3.txt`. Na prvom su studenti sortirani leksi-kografski po imenu rastuće. Na drugom su sortirani po ukupnom broju urađenih zadataka opadajuće, a ukoliko neki studenti imaju isti broj rešenih zadataka sor-tiraju se po dužini imena rastuće. Na trećem spisku kriterijum sortiranja je broj časova na kojima su bili, opadajuće. Ukoliko neki studenti imaju isti broj časova, sortirati ih opadajuće po broju urađenih zadataka, a ukoliko se i on poklapa sor-tirati po prezimenu opadajuće. U datoteci se nalazi ime, prezime studenta, broj

časova na kojima je prisustvovao, kao i ukupan broj uradenih zadataka. Pretpostaviti da studenata neće biti više od 500 i da je za ime studenta dovoljno 20, a za prezime 25 karaktera.

Test 1

AKTIVNOSTI.TXT

```
Aleksandra Cvetic 4 6
Bojan Golubovic 4 3
Dragan Markovic 3 5
Ivana Stankovic 3 1
Marija Stankovic 1 3
Ognjen Peric 1 2
Uros Milic 2 5
Andrija Petrovic 2 5
Anja Ilic 3 1
Lazar Micic 1 3
Nenad Brankovic 2 4
```

DAT1.TXT

```
Studenti sortirani po imenu
leksikografski rastece:
Aleksandra Cvetic 4 6
Andrija Petrovic 2 5
Anja Ilic 3 1
Bojan Golubovic 4 3
Dragan Markovic 3 5
Ivana Stankovic 3 1
Lazar Micic 1 3
Marija Stankovic 1 3
Nenad Brankovic 2 4
Ognjen Peric 1 2
Uros Milic 2 5
```

DAT2.TXT

```
Studenti sortirani po broju zadataka
opadajuće, pa po dužini imena rastece:
Aleksandra Cvetic 4 6
Uros Milic 2 5
Dragan Markovic 3 5
Andrija Petrovic 2 5
Nenad Brankovic 2 4
Lazar Micic 1 3
Bojan Golubovic 4 3
Marija Stankovic 1 3
Ognjen Peric 1 2
Anja Ilic 3 1
Ivana Stankovic 3 1
```

DAT3.TXT

```
Studenti sortirani po prisustvu
opadajuće, pa po broju zadataka,
pa po prezimenima leksikografski
opadajuće:
Aleksandra Cvetic 4 6
Bojan Golubovic 4 3
Dragan Markovic 3 5
Ivana Stankovic 3 1
Anja Ilic 3 1
Andrija Petrovic 2 5
Uros Milic 2 5
Nenad Brankovic 2 4
Marija Stankovic 1 3
Lazar Micic 1 3
Ognjen Peric 1 2
```

Zadatak 1.25 U datoteci `pesme.txt` nalaze se informacije o gledanosti pesama na Youtube-u. Svaki red datoteke sadrži informacije o gledanosti pesama u formatu **izvođač - naslov, broj gledanja**. Napisati program koji učitava informacije o pesmama i vrši sortiranje pesama u zavisnosti od argumenata komandne linije na sledeći način:

- nema opcija, sortiranje se vrši po broju gledanja;
- prisutna je opcija `-i`, sortiranje se vrši po imenima izvođača;
- prisutna je opcija `-n`, sortiranje se vrši po naslovu pesama.

Na standardni izlaz ispisati informacije o pesmama sortiranim na opisani način. Uraditi zadatak bez pravljenja pretpostavki o maksimalnoj dužini imena izvođača i naslova pesme.

<i>Test 1</i>	<i>Test 2</i>	<i>Test 3</i>
POKRETANJE: ./a.out	POKRETANJE: ./a.out -i	POKRETANJE: ./a.out -n
PESME.TXT	PESME.TXT	PESME.TXT
5	5	5
Ana - Nebo, 2342	Ana - Nebo, 2342	Ana - Nebo, 2342
Laza - Oblaci, 29	Laza - Oblaci, 29	Laza - Oblaci, 29
Pera - Ptice, 327	Pera - Ptice, 327	Pera - Ptice, 327
Jelena - Sunce, 92321	Jelena - Sunce, 92321	Jelena - Sunce, 92321
Mika - Kisa, 5341	Mika - Kisa, 5341	Mika - Kisa, 5341
IZLAZ:	IZLAZ:	IZLAZ:
Jelena - Sunce, 92321	Ana - Nebo, 2342	Mika - Kisa, 5341
Mika - Kisa, 5341	Jelena - Sunce, 92321	Ana - Nebo, 2342
Ana - Nebo, 2342	Laza - Oblaci, 29	Laza - Oblaci, 29
Pera - Ptice, 327	Mika - Kisa, 5341	Pera - Ptice, 327
Laza - Oblaci, 29	Pera - Ptice, 327	Jelena - Sunce, 92321

*** Zadatak 1.26** Razmatrajmo dve operacije: operacija U je unos novog broja x , a operacija N određivanje n -tog po veličini od unetih brojeva. Implementirati program koji izvršava ove operacije. Može postojati najviše 100000 operacija unosa, a uneti elementi se mogu ponavljati, pri čemu se i ponavljanja računaju prilikom brojanja. Optimizovati program, ukoliko se zna da neće biti više od 500 različitih unetih brojeva. NAPOMENA: *Brojeve čuvati u sortiranom nizu i svaki naredni element umetati na svoje mesto.*

Primer 1

INTERAKCIJA SA PROGRAMOM:
Unesite niz operacija: U 2 U 0 U 6 U 4 N 1 U 8 N 2 N 5 U 2 N 3 N 5
0 2 8 2 6

*** Zadatak 1.27** Šef u restoranu je neuredan i palačinke koje ispeče ne slaže redom po veličini. Konobar pre serviranja mora da sortira palačinke po veličini, a jedina operacija koju sme da izvodi je da obrne deo palačinki. Na primer, slika 1.1 po kolonama predstavlja naslagane palačinke posle svakog okretanja. Na početku, palačinka veličine 2 je na dnu, iznad nje se redom nalaze najmanja, najveća, itd... Na slici crtica predstavlja mesto iznad koga će konobar okrenuti palačinke. Prvi potez konobara je okretanje palačinki veličine 5, 4 i 3 (prva kolona), i tada će veličine palačinki odozdo nagore biti 2, 1, 3, 4, 5 (druga kolona). Posle još dva okretanja, palačinke će biti složene. Napisati program koji u najviše $2n - 3$ okretanja sortira učitani niz. UPUTSTVO: *Imitirati selection sort i u svakom koraku dovesti jednu palačinku na svoje mesto korišćenjem najviše dva okretanja.*

3	5	2	1
4	4	1_	2
5_	3	3	3
1_	1	4	4
2	2_	5	5

Slika 1.1: *Zadatak 1.27*

Test 1

```

ULAZ:
23 64 123 76 22 7 34 123 54562 12 453 342 5342 42 542 1 3 432 1 32 43

IZLAZ:
1 1 3 7 12 22 23 32 34 42 43 64 76 123 123 342 432 453 542 5342 54562

```

Zadatak 1.28 Za zadatu celobrojnu matricu dimenzije $n \times m$ napisati funkciju koja vrši sortiranje vrsta matrice rastuće na osnovu sume elemenata u vrsti. Napisati potom program koji testira ovu funkciju. Sa standardnog ulaza se prvo unose dimenzije matrice, a zatim redom elementi matrice. Rezultujuću matricu ispisati na standardni izlaz. NAPOMENA: *Koristiti biblioteku za rad sa celobrojnim matricama iz zadatka ??.*

Test 1

```

INTERAKCIJA SA PROGRAMOM:
Unesite dimenzije matrice: 3 2
Unesite elemente matrice po vrstama:
6 -5
-4 3
2 1
Sortirana matrica je:
-4 3
6 -5
2 1

```

Test 2

```

INTERAKCIJA SA PROGRAMOM:
Unesite dimenzije matrice: 4 4
Unesite elemente matrice po vrstama:
34 12 54 642
1 2 3 4
53 2 1 5
54 23 5 671
Sortirana matrica je:
1 2 3 4
53 2 1 5
34 12 54 642
54 23 5 671

```

Zadatak 1.29 Za zadatu kvadratnu matricu dimenzije n napisati funkciju koja sortira kolone matrice opadajuće na osnovu vrednosti prvog elementa u koloni. Napisati program koji testira ovu funkciju. Sa standardnog ulaza se prvo unosi dimenzija matrice, a zatim redom elementi matrice. Rezultujuću matricu ispisati na standardnom izlazu. NAPOMENA: *Koristiti biblioteku za rad sa celobrojnim matricama iz zadatka ??.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
Unesite dimenziju matrice: 2
Unesite elemente matrice po vrstama:
6 -5
-4 3
Sortirana matrica je:
-5 6
3 -4
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
Unesite dimenziju matrice: 4
Unesite elemente matrice po vrstama:
34 12 54 642
1 2 3 4
53 2 1 5
54 23 5 671
Sortirana matrica je:
12 34 54 642
2 1 3 4
2 53 1 5
23 54 5 671
```

1.3 Bibliotečke funkcije pretrage i sortiranja

Zadatak 1.30 Napisati program u kome se prvo inicijalizuje statički niz struktura osoba sa članovima ime i prezime, a zatim se učitava jedan karakter i pronalazi i štampa jedna struktura iz niza osoba čije prezime počinje tim karakterom. Ako takva osoba ne postoji, štampati -1 na standardnom izlazu za greške. Niz struktura ima manje od 100 elemenata i uređen je u rastućem leksikografskom poretku po prezimenima. Pretraživanje niza vršiti bibliotečkom funkcijom `bsearch`. Na primer, niz osoba može da bude inicijalizovan na sledeći način:

```
Osoba niz_osoba[]={{"Mika", "Antic"},
                    {"Dobrica", "Eric"},
                    {"Desanka", "Maksimovic"},
                    {"Dusko", "Radovic"},
                    {"Ljubivoje", "Rsumovic"}};
```

Test 1

```
ULAZ:
R

IZLAZ:
Dusko Radovic
```

Test 2

```
ULAZ:
E

IZLAZ:
Dobrica Eric
```

Test 3

```
ULAZ:
X

IZLAZ:
-1
```

Zadatak 1.31 Napisati program koji ilustruje upotrebu bibliotečkih funkcija za pretraživanje i sortiranje nizova i mogućnost zadavanja različitih kriterijuma sortiranja. Sa standardnog ulaza se unosi dimenzija niza celih brojeva, ne veća od 100, a potom i sami elementi niza. Upotrebom funkcije `qsort` sortirati niz u rastućem poretku, sa standardnog ulaza učitati broj koji se traži u nizu, pa

zatim funkcijama `bsearch` i `lfind` utvrditi da li se zadati broj nalazi u nizu. Na standardni izlaz ispisati odgovarajuću poruku.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
Uneti dimenziju niza: 11
Uneti elemente niza:
5 3 1 6 8 90 34 5 3 432 34
Sortirani niz u rastucem poretku:
1 3 3 5 5 6 8 34 34 90 432
Uneti element koji se trazi u nizu: 34
Binarna pretraga:
Element je nadjen na poziciji 8
Linearna pretraga (lfind):
Element je nadjen na poziciji 7
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
Uneti dimenziju niza: 4
Uneti elemente niza:
4 2 5 7
Sortirani niz u rastucem poretku:
2 4 5 7
Uneti element koji se trazi u nizu: 3
Binarna pretraga:
Elementa nema u nizu!
Linearna pretraga (lfind):
Elementa nema u nizu!
```

Zadatak 1.32 Napisati program koji sa standardnog ulaza učitava dimenziju niza celih brojeva (ne veću od 100), a potom i same elemente niza. Upotrebom funkcije `qsort` sortirati niz u rastućem poretku prema broju delilaca i tako dobijeni niz odštampati na standardni izlaz.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
Uneti dimenziju niza: 10
Uneti elemente niza:
1 2 3 4 5 6 7 8 9 10
Sortirani niz u rastucem
poretku prema broju delilaca
1 2 3 5 7 4 9 6 8 10
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
Uneti dimenziju niza: 1
Uneti elemente niza:
234
Sortirani niz u rastucem
poretku prema broju delilaca
234
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
Uneti dimenziju niza: 0
Uneti elemente niza:
Sortirani niz u rastucem
poretku prema broju
delilaca:
```

Zadatak 1.33 Korišćenjem bibliotečke funkcije `qsort` napisati program koji sortira niz niski po sledećim kriterijumima:

(a) leksikografski,

(b) po dužini.

Niske se učitavaju iz datoteke `niske.txt`. Pretpostaviti da datoteka ne sadrži više od 1000 niski kao i da je svaka niska dužine najviše 30 karaktera. Program prvo leksikografski sortira niz, primenjuje binarnu pretragu (`bsearch`) zarad traženja niske unete sa standardnog ulaza, a potom traži istu nisku koristeći funkciju `lfind` u nizu koji je neposredno pre toga sortiran po dužini. Rezultate svih sortiranja i pretraga ispisati na standardni izlaz.

Primer 1

```
NISKE.TXT
ana petar andjela milos nikola aleksandar ljubica matej milica

INTERAKCIJA SA PROGRAMOM:
Leksikografski sortirane niske:
aleksandar ana andjela ljubica matej milica milos nikola petar
Uneti trazenu nisku: matej
Niska "matej"je pronadjena u nizu na poziciji 4
Niske sortirane po duzini:
ana matej milos petar milica nikola andjela ljubica aleksandar
Niska "matej"je pronadjena u nizu na poziciji 1
```

Zadatak 1.34 Uraditi zadatak 1.33 sa dinamički alociranim niskama i sortiranjem niza pokazivača, umesto niza niski.

Zadatak 1.35 Napisati program koji korišćenjem bibliotečke funkcije `qsort` sortira studente prema broju poena osvojenih na kolokvijumu. Ukoliko više studenata ima isti broj bodova, sortirati ih po prezimenu leksikografski rastuće. Korisnik potom unosi broj bodova i prikazuje mu se jedan od studenata sa tim brojem bodova ili poruka ukoliko nema takvog. Potom, sa standardnog ulaza, unosi se prezime traženog studenta i prikazuje se osoba sa tim prezimenom ili poruka da se nijedan student tako ne preziva. Za pretraživanje koristiti odgovarajuće bibliotečke funkcije. Podaci o studentima čitaju se iz datoteke čije se ime zadaje preko argumenata komandne linije. Za svakog studenta u datoteci postoje ime, prezime i bodovi osvojeni na kolokvijumu. Pretpostaviti da neće biti više od 500 studenata i da je za ime i prezime svakog studenta dovoljno po 20 karaktera.

Primer 1

```
POKRETANJE: ./a.out kolokvijum.txt

ULAZNA DATOTEKA (KOLOKVIJUM.TXT):
Aleksandra Cvetic 15
Bojan Golubovic 30
Dragan Markovic 25
Filip Dukic 20
Ivana Stankovic 25
Marija Stankovic 15
Ognjen Peric 20
Uros Milic 10
Andrija Petrovic 0
Anja Ilic 5
Ivana Markovic 5
Lazar Micic 20
Nenad Brankovic 15

INTERAKCIJA SA PROGRAMOM:
Studenti sortirani po broju poena
opadajuće, pa po prezimenu rastuće:
Bojan Golubovic 30
Dragan Markovic 25
Ivana Stankovic 25
Filip Dukic 20
Lazar Micic 20
Ognjen Peric 20
Nenad Brankovic 15
Aleksandra Cvetic 15
Marija Stankovic 15
Uros Milic 10
Anja Ilic 5
Ivana Markovic 5
Andrija Petrovic 0
Unesite broj bodova: 20
Pronadjen je student sa unetim
brojem bodova: Filip Dukic 20
Unesite prezime: Markovic
Pronadjen je student sa unetim
prezimenom: Dragan Markovic 25
```

Zadatak 1.36 Uraditi zadatak 1.13, ali korišćenjem bibliotečke `qsort` funkcije.

Zadatak 1.37 Napisati program koji sa standardnog ulaza učitava prvo ceo broj n ($n \leq 10$), a zatim niz S od n niski. Maksimalna dužina svake niske je 31 karakter. Sortirati niz S bibliotečkom funkcijom `qsort` i proveriti da li u njemu ima identičnih niski.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
Unesite broj niski: 4
Unesite niske:
prog search sort search
ima
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
Unesite broj niski: 3
Unesite niske:
test kol ispit
nema
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
Unesite broj niski: 5
Unesite niske:
a ab abc abcd abcde
nema
```

Zadatak 1.38 Datoteka `studenti.txt` sadrži spisak studenata. Za svakog studenta poznat je nalog na Alas-u (oblika npr. `mr15125`, `mm14001`), ime, prezime i broj poena. Ni ime, ni prezime, neće biti duže od 20 karaktera. Napisati program koji korišćenjem funkcije `qsort` sortira studente po broju poena opadajuće, ukoliko je prisutna opcija `-p`, ili po nalogu, ukoliko je prisutna opcija `-n`. Studenti se po nalogu sortiraju tako što se sortiraju na osnovu godine, zatim na osnovu smera, i na kraju na osnovu broja indeksa. Sortirane studente upisati u datoteku `izlaz.txt`. Ukoliko je u komandnoj liniji uz opciju `-n` naveden i nalog nekog studenta, funkcijom `bsearch` potražiti i prijaviti broj poena studenta sa tim nalogom.

Test 1

```
POKRETANJE: ./a.out -n mm13321

STUDENTI.TXT
mr14123 Marko Antic 20
mm13321 Marija Radic 12
ml13011 Ivana Mitrovic 19
ml13066 Pera Simic 15
mv14003 Jovan Jovanovic 17

IZLAZ.TXT
ml13011 Ivana Mitrovic 19
ml13066 Pera Simic 15
mm13321 Marija Radic 12
mr14123 Marko Antic 20
mv14003 Jovan Jovanovic 17

IZLAZ:
mm13321 Marija Radic 12
```

Test 2

```
POKRETANJE: ./a.out -p

STUDENTI.TXT
mr14123 Marko Antic 20
mm13321 Marija Radic 12
ml13011 Ivana Mitrovic 19
ml13066 Pera Simic 15
mv14003 Jovan Jovanovic 17

IZLAZ.TXT
mr14123 Marko Antic 20
ml13011 Ivana Mitrovic 19
mv14003 Jovan Jovanovic 17
ml13066 Pera Simic 15
mm13321 Marija Radic 12
```

Zadatak 1.39 Definirati strukturu Datum. Napisati funkciju koja poredi dva datuma hronološki. Potom, napisati i program koji učitava datume iz datoteke koja se zadaje kao prvi argument komandne linije (ne više od 128 datuma), sortira ih pozivajući funkciju `qsort` iz standardne biblioteke i pozivanjem funkcije `bsearch` iz standardne biblioteke proverava da li datumi učitani sa standardnog ulaza postoje među prethodno unetim datumima. Datumi se učitavaju sve do kraja ulaza.

Primer 1

```
POKRETANJE: ./a.out datoteka.txt  
  
DATOTEKA.TXT  
1.1.2013.  
13.12.2016.  
11.11.2011.  
3.5.2015.  
5.2.2009.
```

```
INTERAKCIJA SA PROGRAMOM:  
Unesite sledeci datum: 13.12.2016.  
postoji  
Unesite sledeci datum: 10.5.2015.  
ne postoji  
Unesite sledeci datum: 5.2.2009.  
postoji
```

1.4 Rešenja

Rešenje 1.1

```
1 #include <stdio.h>  
2 #include <stdlib.h>  
3 #include <time.h>  
4 #define MAX 1000000  
5  
6 /* Pri prevodjenju program linkovati sa bibliotekom librt opcijom  
7    -lrt zbog funkcije clock_gettime() */  
8  
9 /* Naredne tri funkcije vrse pretragu. Ukoliko se trazeni  
10    element pronadje u nizu, one vracaju indeks pozicije na kojoj  
11    je element pronadjen. Ovaj indeks je uvek nenegativan. Ako  
12    element nije pronadjen u nizu, funkcije vracaju negativnu  
13    vrednost -1, kao indikator neuspesne pretrage. */  
14  
15 /* Linearna pretraga: Funkcija pretrazuje niz a[] celih brojeva  
16    duzine n, trazeci u njemu prvo pojavljivanje elementa x.  
17    Pretraga se vrsi prostom iteracijom kroz niz. */  
18 int linearna_pretraga(int a[], int n, int x)  
19 {  
20     int i;  
21     for (i = 0; i < n; i++)  
22         if (a[i] == x)  
23             return i;  
24     return -1;  
25 }
```

```

27  /* Binarna pretraga: Funkcija trazi u sortiranom nizu a[] duzine n
28     broj x. Pretraga koristi osobinu sortiranosti niza i u svakoj
29     iteraciji polovi interval pretrage. */
30  int binarna_pretraga(int a[], int n, int x)
31  {
32      int levi = 0;
33      int desni = n - 1;
34      int srednji;
35      /* Dokle god je indeks levi levo od indeksa desni */
36      while (levi <= desni) {
37          /* Srednji indeks je njihova aritmeticka sredina */
38          srednji = (levi + desni) / 2;
39          /* Ako je element sa sredisnjim indeksom veci od x, tada se x
40             mora nalaziti u levom delu niza */
41          if (x < a[srednji])
42              desni = srednji - 1;
43          /* Ako je element sa sredisnjim indeksom manji od x, tada se x
44             mora nalaziti u desnom delu niza */
45          else if (x > a[srednji])
46              levi = srednji + 1;
47          else
48              /* Ako je element sa sredisnjim indeksom jednak x, tada je
49                 broj x pronadjen na poziciji srednji */
49              return srednji;
51      }
52      /* Ako element x nije pronadjen, vraca se -1 */
53      return -1;
54  }
55
56  /* Interpolaciona pretraga: Funkcija trazi u sortiranom nizu a[]
57     duzine n broj x. Pretraga koristi osobinu sortiranosti niza i
58     zasniva se na linearnoj interpolaciji vrednosti koja se trazi
59     vrednostima na krajevima prostora pretrage. */
60  int interpolaciona_pretraga(int a[], int n, int x)
61  {
62      int levi = 0;
63      int desni = n - 1;
64      int srednji;
65      /* Dokle god je indeks levi levo od indeksa desni... */
66      while (levi <= desni) {
67          /* Ako je trazeni element manji od pocetnog ili veci od
68             poslednjeg elementa u delu niza a[levi],...,a[desni], tada
69             on nije u tom delu niza. Ova provera je neophodna, da se ne
70             bi dogodilo da se prilikom izracunavanja indeksa srednji
71             izadje izvan opsega indeksa [levi,desni] */
72          if (x < a[levi] || x > a[desni])
73              return -1;
74          /* U suprotnom, x je izmedju a[levi] i a[desni], pa ako su
75             a[levi] i a[desni] jednaki, tada je jasno da je trazeni broj
76             x jednak ovim vrednostima, pa se vraca indeks levi (ili
77             indeks desni). Ova provera je neophodna, jer bi se u

```

```

79         suprotnom prilikom izracunavanja indeksa srednji pojavilo
           deljenje nulom. */
80     else if (a[levi] == a[desni])
81         return levi;
82     /* Racuna se sredisnji indeks */
83     srednji =
           levi +
84         (int) ((double) (x - a[levi]) / (a[desni] - a[levi]) *
           (desni - levi));
85
86     /* Napomena: Indeks srednji je uvek izmedju levi i desni, ali
           ce verovatno biti blize trazenoj vrednosti nego da je prosto
87     uvek uzimana aritmiticka sredina indeksa levi i desni. Ovo
88     se moze porediti sa pretragom recnika: ako neko trazi rec na
89     slovo 'B', sigurno nece da otvori recnik na polovini, vec
           verovatno negde blize pocetku. */
90
91     /* Ako je element sa indeksom srednji veci od trazenog, tada se
           trazeni element mora nalaziti u levoj polovini niza */
92     if (x < a[srednji])
93         desni = srednji - 1;
94     /* Ako je element sa indeksom srednji manji od trazenog, tada
           se trazeni element mora nalaziti u desnoj polovini niza */
95     else if (x > a[srednji])
96         levi = srednji + 1;
97     else
98         /* Ako je element sa indeksom srednji jednak trazenom, onda
           se pretraga zavrшава na poziciji srednji */
99         return srednji;
100 }
101
102 /* U slucaju neuspesne pretrage vraca se -1 */
103 return -1;
104 }
105
106
107
108
109 int main(int argc, char **argv)
110 {
111     int a[MAX];
112     int n, i, x;
113     struct timespec vreme1, vreme2, vreme3, vreme4, vreme5, vreme6;
114     FILE *f;
115     /* Provera argumenata komandne linije */
116     if (argc != 3) {
117         fprintf(stderr,
118             "Greska: Program se poziva sa %s dim_niza broj\n",
119             argv[0]);
120         exit(EXIT_FAILURE);
121     }
122
123     /* Dimenzija niza */
124     n = atoi(argv[1]);
125     if (n > MAX || n <= 0) {
126         fprintf(stderr, "Greska: Dimenzija niza neodgovarajuca\n");
127         exit(EXIT_FAILURE);
128     }
129 }

```

```

131  /* Broj koji se trazi */
132  x = atoi(argv[2]);
133  /* Elementi niza se generisu slucajno, tako da je svaki sledeci
134     veci od prethodnog. Funkcija srandom() inicijalizuje pocetnu
135     vrednost sa kojom se krece u izracunavanje sekvence
136     pseudo-slucajnih brojeva. Kako generisani niz ne bi uvek bio
137     isti, ova vrednost se postavlja na tekuce vreme u sekundama od
138     Nove godine 1970, tako da je za svako sledece pokretanje
139     programa (u vremenskim intervalima vecim od jedne sekunde) ove
140     vrednost drugacija. random()%100 vraca brojeve izmedju 0 i 99 */
141  srandom(time(NULL));
142  for (i = 0; i < n; i++)
143      a[i] = i == 0 ? random() % 100 : a[i - 1] + random() % 100;
144  /* Linearna pretraga */
145  printf("Linearna pretraga:\n");
146  /* Vreme proteklo od Nove godine 1970 */
147  clock_gettime(CLOCK_REALTIME, &vreme1);
148  i = linearna_pretraga(a, n, x);
149  /* Novo vreme i razlika sa prvim predstavlja vreme utroseno za
150     linearnu pretragu */
151  clock_gettime(CLOCK_REALTIME, &vreme2);
152  if (i == -1)
153      printf("Element nije u nizu\n");
154  else
155      printf("Element je u nizu na poziciji %d\n", i);
156  /* Binarna pretraga */
157  printf("Binarna pretraga:\n");
158  clock_gettime(CLOCK_REALTIME, &vreme3);
159  i = binarna_pretraga(a, n, x);
160  clock_gettime(CLOCK_REALTIME, &vreme4);
161  if (i == -1)
162      printf("Element nije u nizu\n");
163  else
164      printf("Element je u nizu na poziciji %d\n", i);
165  /* Interpolaciona pretraga */
166  printf("Interpolaciona pretraga:\n");
167  clock_gettime(CLOCK_REALTIME, &vreme5);
168  i = interpolaciona_pretraga(a, n, x);
169  clock_gettime(CLOCK_REALTIME, &vreme6);
170  if (i == -1)
171      printf("Element nije u nizu\n");
172  else
173      printf("Element je u nizu na poziciji %d\n", i);
174  /* Podaci o izvršavanju programa bivaju upisani u log */
175  if ((f = fopen("vremena.txt", "a")) == NULL) {
176      fprintf(stderr, "Greska: Neuspesno otvaranje log datoteke.\n");
177      exit(EXIT_FAILURE);
178  }
179
180  fprintf(f, "Dimenzija niza: %d\n", n);
181  fprintf(f, "\tLinearna:%10ld ns\n",

```

```

183         (vreme2.tv_sec - vreme1.tv_sec) * 1000000000 +
            vreme2.tv_nsec - vreme1.tv_nsec);
185     fprintf(f, "\tBinarna: %19ld ns\n",
            (vreme4.tv_sec - vreme3.tv_sec) * 1000000000 +
            vreme4.tv_nsec - vreme3.tv_nsec);
187     fprintf(f, "\tInterpolaciona: %12ld ns\n\n",
            (vreme6.tv_sec - vreme5.tv_sec) * 1000000000 +
            vreme6.tv_nsec - vreme5.tv_nsec);
189     /* Zatvara se datoteka */
191     fclose(f);
193     exit(EXIT_SUCCESS);
}

```

Rešenje 1.2

```

#include <stdio.h>
2 #include <stdlib.h>

4 #define MAX 1024

6 /* Rekurzivna linearna pretraga od pocetka niza */
int linearna_pretraga_r1(int a[], int n, int x)
8 {
    int tmp;
    /* Izlaz iz rekurziije */
    if (n <= 0)
10         return -1;
    /* Ako je prvi element trazeni */
12     if (a[0] == x)
        return 0;
14     /* Pretraga ostatka niza */
    tmp = linearna_pretraga_r1(a + 1, n - 1, x);
16     return tmp < 0 ? tmp : tmp + 1;
18 }

20 /* Rekurzivna linearna pretraga od kraja niza */
22 int linearna_pretraga_r2(int a[], int n, int x)
{
    /* Izlaz iz rekurziije */
24     if (n <= 0)
        return -1;
26     /* Ako je poslednji element trazeni */
    if (a[n - 1] == x)
28         return n - 1;
    /* Pretraga ostatka niza */
30     return linearna_pretraga_r2(a, n - 1, x);
32 }

34 /* Rekurzivna binarna pretraga */
int binarna_pretraga_r(int a[], int l, int d, int x)

```



```

36 {
37     int srednji;
38     if (l > d)
39         return -1;
40     /* Sredisnja pozicija na kojoj se trazi vrednost x */
41     srednji = (l + d) / 2;
42     /* Ako je element na sredisnjoj poziciji trazeni */
43     if (a[srednji] == x)
44         return srednji;
45     /* Ako je trazeni broj veci od broja na sredisnjoj poziciji,
46        pretrazuje se desna polovina niza */
47     if (a[srednji] < x)
48         return binarna_pretraga_r(a, srednji + 1, d, x);
49     /* Ako je trazeni broj manji od broja na sredisnjoj poziciji,
50        pretrazuje se leva polovina niza */
51     else
52         return binarna_pretraga_r(a, l, srednji - 1, x);
53 }
54
55 /* Rekurzivna interpolaciona pretaga */
56 int interpolaciona_pretraga_r(int a[], int l, int d, int x)
57 {
58     int p;
59     /* Ako je trazeni element manji od prvog ili veci od poslednjeg */
60     if (x < a[l] || x > a[d])
61         return -1;
62     /* Ako je ostao jedan element u delu niza koji se pretrazuje */
63     if (a[d] == a[l])
64         return l;
65     /* Pozicija na kojoj se trazi vrednost x */
66     p = l + (d - l) * (x - a[l]) / (a[d] - a[l]);
67     if (a[p] == x)
68         return p;
69     /* Pretraga sufiksa niza */
70     if (a[p] < x)
71         return interpolaciona_pretraga_r(a, p + 1, d, x);
72     /* Pretraga prefiksa niza */
73     else
74         return interpolaciona_pretraga_r(a, l, p - 1, x);
75 }
76
77 int main()
78 {
79     int a[MAX], x, i, indeks;
80
81     /* Ucitava se trazeni broj */
82     printf("Unesite trazeni broj: ");
83     scanf("%d", &x);
84
85     /* Ucitavaju se elementi niza sve do kraja ulaza - ocekuje se da
86        korisnik pritisne CTRL+D za naznaku kraja */
87     i = 0;

```

```

88     printf("Unesite sortiran niz elemenata: ");
    while (i < MAX && scanf("%d", &a[i]) == 1) {
90         if (i > 0 && a[i] < a[i - 1]) {
            fprintf(stderr, "Greska: Elementi moraju biti uneseni ");
92             fprintf(stderr, "u neopadajućem poretku\n");
            exit(EXIT_FAILURE);
94         }
        i++;
96     }

98     /* Rezultati linearnih pretraga */
    printf("Linearna pretraga\n");
    indeks = linearna_pretraga_r1(a, i, x);
100    if (indeks == -1)
102        printf("Element se ne nalazi u nizu.\n");
    else
104        printf("Pozicija elementa je %d.\n", indeks);
    indeks = linearna_pretraga_r2(a, i, x);
106    if (indeks == -1)
        printf("Element se ne nalazi u nizu.\n");
108    else
        printf("Pozicija elementa je %d.\n", indeks);
110

    /* Rezultati binarna pretrage */
112    printf("Binarna pretraga\n");
    indeks = binarna_pretraga_r(a, 0, i - 1, x);
114    if (indeks == -1)
        printf("Element se ne nalazi u nizu.\n");
116    else
        printf("Pozicija elementa je %d.\n", indeks);
118

    /* Rezultati interpolacione pretrage */
120    printf("Interpolaciona pretraga\n");
    indeks = interpolaciona_pretraga_r(a, 0, i - 1, x);
122    if (indeks == -1)
        printf("Element se ne nalazi u nizu.\n");
124    else
        printf("Pozicija elementa je %d.\n", indeks);
126

    exit(EXIT_SUCCESS);
128 }

```

Rešenje 1.3

```

#include <stdio.h>
2 #include <stdlib.h>
#include <string.h>
4
#define MAX_STUDENATA 128
6 #define MAX_DUZINA 16

```

```

8  /* 0 svakom studentu postoje 3 informacije i one su objedinjene u
   strukturi kojom se predstavlja svaki student. */
10 typedef struct {
   /* Indeks mora biti tipa long jer su podaci u datoteci preveliki
   za int, npr. 20140123 */
12     long indeks;
14     char ime[MAX_DUZINA];
   char prezime[MAX_DUZINA];
16 } Student;

18 /* Učitani niz studenata će biti sortiran rastuće prema indeksu, jer
   su studenti u datoteci već sortirani. Iz tog razloga pretraga po
20     indeksu se vrši binarno, dok pretraga po prezimenu mora linearno,
   jer nema garancije da postoji uređenje po prezimenu. */

22 /* Funkcija traži u sortiranom nizu studenata a[] dužine n studenta
   sa indeksom x i vraća indeks pozicije nadjenog člana niza ili -1,
   ako element nije pronađen. */
24 int binarna_pretraga(Student a[], int n, long x)
   {
26     int levi = 0;
     int desni = n - 1;
30     int srednji;
     /* Dokle god je indeks levi levo od indeksa desni */
32     while (levi <= desni) {
         /* Racuna se srednja pozicija */
34         srednji = (levi + desni) / 2;
         /* Ako je indeks studenta na toj poziciji veći od traženog,
36         tada se traženi indeks mora nalaziti u levoj polovini niza */
         if (x < a[srednji].indeks)
38             desni = srednji - 1;
         /* Ako je pak manji od traženog, tada se on mora nalaziti u
40         desnoj polovini niza */
         else if (x > a[srednji].indeks)
42             levi = srednji + 1;
         else
44             /* Ako je jednak traženom indeksu x, tada je pronađen
             student sa traženom indeksom na poziciji srednji */
46             return srednji;
     }
48     /* Ako nije pronađen, vraća se -1 */
     return -1;
50 }

52 /* Linearnom pretragom niza studenata traži se prezime x */
int linearna_pretraga(Student a[], int n, char x[])
54 {
     int i;
56     for (i = 0; i < n; i++)
         /* Poredi se prezime i-tog studenta i poslatog x */
58         if (strcmp(a[i].prezime, x) == 0)
             return i;
}

```

```

60     return -1;
61 }
62
63 /* Main funkcija mora imati argumente jer se ime datoteke i opcija
64    prosledjuju kao argumenti komandne linije */
65 int main(int argc, char *argv[])
66 {
67     Student dosije[MAX_STUDENATA];
68     FILE *fin = NULL;
69     int i;
70     int br_studenata = 0;
71     long trazen_indeks = 0;
72     char trazeno_prezime[MAX_DUZINA];
73     int bin_pretraga;
74
75     /* Provera da li je korisnik prilikom poziva programa prosledio
76        ime datoteke sa informacijama o studentima i opciju pretrage */
77     if (argc != 3) {
78         fprintf(stderr,
79             "Greska: Program se poziva sa %s ime_datoteke opcija\n",
80             argv[0]);
81         exit(EXIT_FAILURE);
82     }
83
84     /* Provera prosledjene opcije */
85     if (strcmp(argv[2], "-indeks") == 0)
86         bin_pretraga = 1;
87     else if (strcmp(argv[2], "-prezime") == 0)
88         bin_pretraga = 0;
89     else {
90         fprintf(stderr,
91             "Greska: Opcija mora biti -indeks ili -prezime\n");
92         exit(EXIT_FAILURE);
93     }
94
95     /* Otvara se datoteka */
96     fin = fopen(argv[1], "r");
97     if (fin == NULL) {
98         fprintf(stderr,
99             "Greska: Neuspesno otvaranje datoteke %s za citanje\n",
100             argv[1]);
101         exit(EXIT_FAILURE);
102     }
103
104     /* Cita se sve dok postoji red sa informacijama o studentu */
105     i = 0;
106     while (1) {
107         if (i == MAX_STUDENATA)
108             break;
109         if (fscanf
110             (fin, "%ld %s %s", &dosije[i].indeks, dosije[i].ime,
111              dosije[i].prezime) != 3)

```

```

112         break;
113         i++;
114     }
115     br_studenata = i;
116
117     /* Nakon citanja, datoteka vise nije neophodna i zatvara se. */
118     fclose(fin);
119
120     /* Pretraga po indeksu */
121     if (bin_pretraga) {
122         /* Unos indeksa koji se binarno trazi u nizu */
123         printf("Unesite indeks studenta cije informacije zelite: ");
124         scanf("%ld", &trazen_indeks);
125         i = binarna_pretraga(dosije, br_studenata, trazen_indeks);
126         /* Rezultat binarne pretrage */
127         if (i == -1)
128             printf("Ne postoji student sa indeksom %ld\n", trazen_indeks);
129         else
130             printf("Indeks: %ld, Ime i prezime: %s %s\n",
131                   dosije[i].indeks, dosije[i].ime, dosije[i].prezime);
132     }
133     /* Pretraga po prezimenu */
134     else {
135         /* Unos prezimena koje se linearno trazi u nizu */
136         printf("Unesite prezime studenta cije informacije zelite: ");
137         scanf("%s", trazeno_prezime);
138         i = linearna_pretraga(dosije, br_studenata, trazeno_prezime);
139         /* Rezultat linearne pretrage */
140         if (i == -1)
141             printf("Ne postoji student sa prezimenom %s\n",
142                   trazeno_prezime);
143         else
144             printf("Indeks: %ld, Ime i prezime: %s %s\n",
145                   dosije[i].indeks, dosije[i].ime, dosije[i].prezime);
146     }
147
148     exit(EXIT_SUCCESS);
149 }

```

Rešenje 1.4

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #define MAX_STUDENATA 128
6  #define MAX_DUZINA 16
7
8  typedef struct {
9      long indeks;
10     char ime[MAX_DUZINA];

```

```

11     char prezime[MAX_DUZINA];
12 } Student;
13
14 int binarna_pretraga_rekurzivna(Student a[], int levi, int desni,
15                                 long x)
16 {
17     /* Ako je pozicija elementa na levom kraju veca od pozicije
18        elementa na desnom kraju dela niza koji se pretrazuje, onda se
19        zapravo pretrazuje prazan deo niza. U praznom delu niza nema
20        trazenog elementa pa se vraca -1 */
21     if (levi > desni)
22         return -1;
23     /* Racuna se pozicija srednjeg elementa */
24     int srednji = (levi + desni) / 2;
25     /* Da li je srednji bas onaj trazeni */
26     if (a[srednji].indeks == x) {
27         return srednji;
28     }
29     /* Ako je trazeni indeks manji od indeksa studenta na srednjoj
30        poziciji, onda se pretraga nastavlja u levoj polovini niza,
31        jer je poznato da je niz sortiran po indeksu u rastucem
32        poretku. */
33     if (x < a[srednji].indeks)
34         return binarna_pretraga_rekurzivna(a, levi, srednji - 1, x);
35     /* Inace ga treba traziti u desnoj polovini */
36     else
37         return binarna_pretraga_rekurzivna(a, srednji + 1, desni, x);
38 }
39
40 int linearna_pretraga_rekurzivna_v2(Student a[], int n, char x[])
41 {
42     /* Ako je niz prazan, vraca se -1 */
43     if (n == 0)
44         return -1;
45     /* Kako se trazi prvi student sa trazanim prezimenom, pocinje se
46        sa prvim studentom u nizu. */
47     if (strcmp(a[0].prezime, x) == 0)
48         return 0;
49     int i = linearna_pretraga_rekurzivna_v2(a + 1, n - 1, x);
50     return i >= 0 ? 1 + i : -1;
51 }
52
53 int linearna_pretraga_rekurzivna(Student a[], int n, char x[])
54 {
55     /* Ako je niz prazan, vraca se -1 */
56     if (n == 0)
57         return -1;
58     /* Ako se trazi poslednji student sa trazanim prezimenom, pocinje
59        se sa poslednjim studentom u nizu. */
60     if (strcmp(a[n - 1].prezime, x) == 0)
61         return n - 1;
62     return linearna_pretraga_rekurzivna(a, n - 1, x);

```

```

63 }

65 /* Main funkcija mora imati argumente jer se ime datoteke i opcija
   prosledjuju kao argumenti komandne linije */
67 int main(int argc, char *argv[])
68 {
69     Student dosije[MAX_STUDENATA];
70     FILE *fin = NULL;
71     int i;
72     int br_studenata = 0;
73     long trazen_indeks = 0;
74     char trazeno_prezime[MAX_DUZINA];
75     int bin_pretraga;

77     /* Provera da li je korisnik prilikom poziva programa prosledio
       ime datoteke sa informacijama o studentima i opciju pretrage */
79     if (argc != 3) {
80         fprintf(stderr,
81             "Greska: Program se poziva sa %s ime_datoteke opcija\n",
82             argv[0]);
83         exit(EXIT_FAILURE);
84     }

85     /* Provera prosledjene opcije */
86     if (strcmp(argv[2], "-indeks") == 0)
87         bin_pretraga = 1;
88     else if (strcmp(argv[2], "-prezime") == 0)
89         bin_pretraga = 0;
90     else {
91         fprintf(stderr,
92             "Greska: Opcija mora biti -indeks ili -prezime\n");
93         exit(EXIT_FAILURE);
94     }

95     /* Otvara se datoteka */
96     fin = fopen(argv[1], "r");
97     if (fin == NULL) {
98         fprintf(stderr,
99             "Greska: Neuspesno otvaranje datoteke %s za citanje\n",
100             argv[1]);
101         exit(EXIT_FAILURE);
102     }

103     /* Cita se sve dok postoji red sa informacijama o studentu */
104     i = 0;
105     while (1) {
106         if (i == MAX_STUDENATA)
107             break;
108         if (fscanf
109             (fin, "%ld %s %s", &dosije[i].indeks, dosije[i].ime,
110              dosije[i].prezime) != 3)
111             break;
112     }

```

```

115     i++;
116 }
117 br_studenata = i;

119 /* Nakon citanja, datoteka vise nije neophodna i zatvara se. */
120 fclose(fin);

121
122 /* Pretraga po indeksu */
123 if (bin_pretraga) {
124     /* Unos indeksa koji se binarno trazi u nizu */
125     printf("Unesite indeks studenta cije informacije želite: ");
126     scanf("%ld", &trazen_indeks);
127     i = binarna_pretraga_rekurzivna(dosije, 0, br_studenata,
128                                     trazen_indeks);

129     /* Rezultat binarne pretrage */
130     if (i == -1)
131         printf("Ne postoji student sa indeksom %ld\n", trazen_indeks);
132     else
133         printf("Indeks: %ld, Ime i prezime: %s %s\n",
134               dosije[i].indeks, dosije[i].ime, dosije[i].prezime);
135 }
136 /* Pretraga po prezimenu */
137 else {
138     /* Unos prezimena koje se linearno trazi u nizu */
139     printf("Unesite prezime studenta cije informacije želite: ");
140     scanf("%s", trazeno_prezime);
141     i = linearna_pretraga_rekurzivna_v2(dosije, br_studenata,
142                                         trazeno_prezime);

143     /* Rezultat linearne pretrage */
144     if (i == -1)
145         printf("Ne postoji student sa prezimenom %s\n",
146               trazeno_prezime);
147     else
148         printf("Indeks: %ld, Ime i prezime: %s %s\n",
149               dosije[i].indeks, dosije[i].ime, dosije[i].prezime);
150 }
151
152 exit(EXIT_SUCCESS);
153 }

```

Rešenje 1.5

```

1 #include <stdio.h>
2 #include <string.h>
3 #include <math.h>
4 #include <stdlib.h>
5
6 #define MAX 1024
7
8 /* Struktura koja opisuje tacku u ravni */
9 typedef struct Tacka {

```



```

11     float x;
12     float y;
13 } Tacka;

14
15 /* Funkcija koja racuna rastojanje zadate tacke od koordinatnog
16    pocetka (0,0) */
17 float rastojanje(Tacka A)
18 {
19     return sqrt(A.x * A.x + A.y * A.y);
20 }

21 /* Funkcija koja pronalazi tacku najblizu koordinatnom pocetku u
22    nizu zadatih tacaka t dimenzije n */
23 Tacka najbliza_koordinatnom(Tacka t[], int n)
24 {
25     Tacka najbliza;
26     int i;
27     najbliza = t[0];
28     for (i = 1; i < n; i++) {
29         if (rastojanje(t[i]) < rastojanje(najbliza)) {
30             najbliza = t[i];
31         }
32     }
33     return najbliza;
34 }

35
36 /* Funkcija koja pronalazi tacku najblizu x osi u nizu zadatih
37    tacaka t dimenzije n */
38 Tacka najbliza_x_osi(Tacka t[], int n)
39 {
40     Tacka najbliza;
41     int i;
42     najbliza = t[0];
43     for (i = 1; i < n; i++) {
44         if (fabs(t[i].x) < fabs(najbliza.x)) {
45             najbliza = t[i];
46         }
47     }
48     return najbliza;
49 }

50
51 /* Funkcija koja pronalazi tacku najblizu y osi u nizu zadatih
52    tacaka t dimenzije n */
53 Tacka najbliza_y_osi(Tacka t[], int n)
54 {
55     Tacka najbliza;
56     int i;
57     najbliza = t[0];
58     for (i = 1; i < n; i++) {
59         if (fabs(t[i].y) < fabs(najbliza.y)) {
60             najbliza = t[i];
61         }
62     }

```

```

    }
63     return najbliza;
    }
65
66 int main(int argc, char *argv[])
67 {
    FILE *ulaz;
69     Tacka tacke[MAX];
    Tacka najbliza;
71     int i, n;

73     /* Ocekuje se da korisnik prosledi barem ime izvrsnog programa i
        ime datoteke sa tackama */
75     if (argc < 2) {
        fprintf(stderr,
77             "Greska: Programa se poziva sa %s ime_datoteke\n",
                argv[0]);
79         exit(EXIT_FAILURE);
    }

81
    /* Otvara se datoteka za citanje */
83     ulaz = fopen(argv[1], "r");
    if (ulaz == NULL) {
85         fprintf(stderr, "Greska: Neuspesno otvaranje datoteke %s!\n",
            argv[1]);
87         exit(EXIT_FAILURE);
    }

89
    /* Sve dok ima tacaka u datoteci, one bivaju smestane u niz sa
        tackama; i predstavlja indeks tekuce tacke */
91     i = 0;
93     while (fscanf(ulaz, "%f %f", &tacke[i].x, &tacke[i].y) == 2) {
        i++;
95     }
    n = i;

97
    /* Proverava se koji su dodatni argumenti komandne linije. */
99     /* Ako nema dodatnih argumenata */
    if (argc == 2)
101         /* Trazi se najbliza tacka u odnosu na koordinatni pocetak */
        najbliza = najbliza_koordinatnom(tacke, n);
103     /* Inace proverava se koji je dodatni argument prosledjen. Ako je
        u pitanju opcija -x */
105     else if (strcmp(argv[2], "-x") == 0)
        /* Racuna se rastojanje u odnosu na x osu */
107         najbliza = najbliza_x_osi(tacke, n);
    /* Ako je u pitanju opcija -y */
109     else if (strcmp(argv[2], "-y") == 0)
        /* Racuna se rastojanje u odnosu na y osu */
111         najbliza = najbliza_y_osi(tacke, n);
    else {
113         /* Ako nije zadata opcija -x ili -y, ispisuje se obavestenje za

```

```

115     korisnika i prekida se izvršavanje programa */
    fprintf(stderr, "Greska: Pogresna opcija\n");
    exit(EXIT_FAILURE);
117 }

119 /* Stampaju se koordinate trazene tacke */
    printf("%g %g\n", najbliza.x, najbliza.y);
121
    /* Zatvara se datoteka */
123    fclose(ulaz);

125    exit(EXIT_SUCCESS);
}

```

Rešenje 1.6

```

#include <stdio.h>
2  #include <math.h>

4  /* Tacnost */
#define EPS 0.001

6
int main()
8  {
    /* Za interval [0, 2] leva granica je 0, a desna 2 */
10    double l = 0, d = 2, s;

12    /* Sve dok se ne pronadje trazena vrednost argumenta */
    while (1) {
14        /* Polovi se interval */
        s = (l + d) / 2;
16        /* Ako je apsolutna vrednost kosinusa u ovoj tacki manja od
           zadate tacnosti, prekida se pretraga */
18        if (fabs(cos(s)) < EPS)
            break;
20        /* Ako je nula u levom delu intervala, nastavlja se pretraga na
           [l, s] */
22        if (cos(l) * cos(s) < 0)
            d = s;
24        else
            /* Inace, na intervalu [s, d] */
26            l = s;
    }

28
    /* Stampa se vrednost trazene tacke */
30    printf("%g\n", s);

32    return 0;
}

```

Rešenje 1.7

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <math.h>
5
6  int main(int argc, char **argv)
7  {
8      double l, d, s, epsilon;
9
10     char ime_funkcije[6];
11
12     /* Pokazivac na funkciju koja ima jedan argument tipa double i
13        povratnu vrednost istog tipa */
14     double (*fp) (double);
15
16     /* Ako korisnik nije uneo argument, prijavljuje se greska */
17     if (argc != 2) {
18         fprintf(stderr,
19                 "Greska: Program se poziva sa %s ime_funkcije\n",
20                 argv[0]);
21         exit(EXIT_FAILURE);
22     }
23
24     /* Niska ime_funkcije sadrzi ime trazene funkcije koja je
25        navedena u komandnoj liniji */
26     strcpy(ime_funkcije, argv[1]);
27
28     /* Inicijalizuje se pokazivac na funkciju koja se tabelira */
29     if (strcmp(ime_funkcije, "sin") == 0)
30         fp = &sin;
31     else if (strcmp(ime_funkcije, "cos") == 0)
32         fp = &cos;
33     else if (strcmp(ime_funkcije, "tan") == 0)
34         fp = &tan;
35     else if (strcmp(ime_funkcije, "atan") == 0)
36         fp = &atan;
37     else if (strcmp(ime_funkcije, "asin") == 0)
38         fp = &asin;
39     else if (strcmp(ime_funkcije, "log") == 0)
40         fp = &log;
41     else if (strcmp(ime_funkcije, "log10") == 0)
42         fp = &log10;
43     else {
44         fprintf(stderr,
45                 "Greska: Program ne podrzava trazenu funkciju!\n");
46         exit(EXIT_SUCCESS);
47     }
48
49     printf("Unesite krajeve intervala: ");
50     scanf("%lf %lf", &l, &d);
```

```

52  if ((*fp) (l) * (*fp) (d) >= 0) {
    fprintf(stderr, "Greska: %s na intervalu ", ime_funkcije);
54  fprintf(stderr, "[%g, %g] ne zadovoljava uslove\n", l, d);
    exit(EXIT_FAILURE);
56  }

58  printf("Unesite preciznost: ");
    scanf("%lf", &epsilon);

60

/* Sve dok se ne pronadje trazena vrednost argumenta */
62  while (1) {
    /* Polovi se interval */
64    s = (l + d) / 2;
    /* Ako je apsolutna vrednost trazene funkcije u ovoj tacki
66    manja od zadate tacnosti, prekida se pretraga */
    if (fabs((*fp) (s)) < epsilon) {
68        break;
    }
70    /* Ako je nula u levom delu intervala, nastavlja se pretraga na
       [l, s] */
72    if ((*fp) (l) * (*fp) (s) < 0)
        d = s;
74    else
        /* Inace, na intervalu [s, d] */
76        l = s;
    }

78    /* Stampa se vrednost trazene tacke */
80    printf("%g\n", s);

82    exit(EXIT_SUCCESS);
}

```

Rešenje 1.8

```

#include <stdio.h>
2  #include <stdlib.h>

4  #define MAX 256

6  int prvi_veci_od_nule(int niz[], int n)
{
8    /* Granice pretrage */
    int l = 0, d = n - 1;
10   int s;
    /* Sve dok je leva manja od desne granice */
12   while (l <= d) {
        /* Racuna se sredisnja pozicija */
14       s = (l + d) / 2;
        /* Ako je broj na toj poziciji veci od nule, a eventualni

```

```

16         njegov prethodnik manji ili jednak nuli, pretraga je
           završena */
18     if (niz[s] > 0 && ((s > 0 && niz[s - 1] <= 0) || s == 0))
           return s;
20     /* U slučaju broja manjeg ili jednakog nuli, pretražuje se
           desna polovina niza */
22     if (niz[s] <= 0)
           l = s + 1;
24     /* A inace, leva polovina */
           else
26         d = s - 1;
           }
28     return -1;
   }

30 int main()
31 {
32     int niz[MAX];
33     int n = 0;

34     /* Unos niza */
35     while (scanf("%d", &niz[n]) == 1)
36         n++;

37     /* Stampa se rezultat */
38     printf("%d\n", prvi_veci_od_nule(niz, n));

39     return 0;
40 }

```

Rešenje 1.9

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define MAX 256
5
6 int prvi_manji_od_nule(int niz[], int n)
7 {
8     /* Granice pretrage */
9     int l = 0, d = n - 1;
10    int s;
11    /* Sve dok je leva manja od desne granice */
12    while (l <= d) {
13        /* Racuna se sredisnja pozicija */
14        s = (l + d) / 2;
15        /* Ako je broj na toj poziciji manji od 0, a eventualni njegov
           prethodnik veci ili jednak 0, pretraga se završava */
16        if (niz[s] < 0 && ((s > 0 && niz[s - 1] >= 0) || s == 0))
17            return s;
18        /* Ako je broj veci ili jednak 0, pretražuje se desna polovina

```

```

    niza */
21     if (niz[s] >= 0)
        l = s + 1;
23     /* A inace leva */
        else
25         d = s - 1;
    }
27     return -1;
}

29
31 int main()
{
    int niz[MAX];
33     int n = 0;

    /* Unos niza */
35     while (scanf("%d", &niz[n]) == 1)
37         n++;

    /* Stampa se rezultat */
39     printf("%d\n", prvi_manji_od_nule(niz, n));
41
43     return 0;
}

```

Rešenje 1.10

```

1  #include <stdio.h>
   #include <stdlib.h>

3
   unsigned int logaritam_a(unsigned int x)
5  {
   /* Izlaz iz rekurzije */
7   if (x == 1)
       return 0;
   /* Rekurzivni korak */
9   return 1 + logaritam_a(x >> 1);
11 }

13 unsigned int logaritam_b(unsigned int x)
{
15     /* Binarnom pretragom se trazi jedinica u binarnom zapisu broja x
       najvece vaznosti, tj. najlevlja. Pretragu se vrši od pozicije 0
       do 31 */
17     int d = 0, l = sizeof(unsigned int) * 8 - 1;
    int s;
19     /* Sve dok je desna granica pretrage desnije od leve */
    while (d <= l) {
21         /* Racuna se sredisnja pozicija */
        s = (l + d) / 2;
23         /* Proverava se da li je na toj poziciji trazena jedinica */

```

```

25     if ((1 << s) <= x && (1 << (s + 1)) > x)
26         return s;
27     /* Pretraga desne polovine binarnog zapisa */
28     if ((1 << s) > x)
29         l = s - 1;
30     /* Pretraga leve polovine binarnog zapisa */
31     else
32         d = s + 1;
33 }
34 return s;
35 }
36
37 int main()
38 {
39     unsigned int x;
40
41     /* Unos podatka */
42     scanf("%u", &x);
43
44     /* Provera da li je uneti broj pozitivan */
45     if (x == 0) {
46         fprintf(stderr, "Greska: Logaritam od nule nije definisan\n");
47         exit(EXIT_FAILURE);
48     }
49
50     /* Ispis povratnih vrednosti funkcija */
51     printf("%u %u\n", logaritam_a(x), logaritam_b(x));
52
53     exit(EXIT_SUCCESS);
54 }

```

Rešenje 1.12

sort.h

```

1  #ifndef _SORT_H_
2  #define _SORT_H_ 1
3
4  /* Selection sort: Funkcija sortira niz celih brojeva metodom
5     sortiranja izborom. Ideja algoritma je sledeca: U svakoj
6     iteraciji pronalazi se najmanji element i premesta se na pocetak
7     niza. Dakle, u prvoj iteraciji, pronalazi se najmanji element, i
8     dovodi na nulto mesto u nizu. U i-toj iteraciji najmanjih i-1
9     elemenata su vec na svojim pozicijama, pa se od elemenata sa
10    indeksima od i do n-1 trazi najmanji, koji se dovodi na i-tu
11    poziciju. */
12 void selection_sort(int a[], int n);
13
14 /* Insertion sort: Funkcija sortira niz celih brojeva metodom
15    sortiranja umetanjem. Ideja algoritma je sledeca: neka je na

```



```

16 pocetku i-te iteracije niz prvih i elemenata
18 (a[0],a[1],...,a[i-1]) sortirano. U i-toj iteraciji treba
20 element a[i] umetnuti na pravu poziciju medju prvih i elemenata
22 tako da se dobije niz duzine i+1 koji je sortiran. Ovo se radi
24 tako sto se i-ti element najpre uporedi sa njegovim prvim levim
26 susedom (a[i-1]). Ako je a[i] vece, tada je on vec na pravom
    mestu, i niz a[0],a[1],...,a[i] je sortiran, pa se moze preci na
    sledecu iteraciju. Ako je a[i-1] vece, tada se zamenjuju a[i] i
    a[i-1], a zatim se proverava da li je potrebno dalje
    potiskivanje elementa u levo, poredeci ga sa njegovim novim
    levim susedom. Ovim uzastopnim premestanjem se a[i] umece na
    pravo mesto u nizu. */
28 void insertion_sort(int a[], int n);

30 /* Bubble sort: Funkcija sortira niz celih brojeva metodom
32 mehurica. Ideja algoritma je sledeca: prolazi se kroz niz redom
34 poredeci susedne elemente, i pri tom ih zamenjujuci ako su u
    pogresnom poretku. Ovim se najveći element poput mehurica
    istiskuje na "povrsinu", tj. na krajnju desnu poziciju. Nakon
    toga je potrebno ovaj postupak ponoviti nad nizom
36 a[0],...,a[n-2], tj. nad prvih n-1 elemenata niza bez poslednjeg
    koji je postavljen na pravu poziciju. Nakon toga se isti
    postupak ponavlja nad sve kracim i kracim prefiksima niza, cime
    se jedan po jedan istiskuju elementi na svoje prave pozicije. */
40 void bubble_sort(int a[], int n);

42 /* Selsort: Ovaj algoritam je jednostavno prosirenje sortiranja
44 umetanjem koje dopusta direktnu razmenu udaljenih elemenata.
    Prosirenje se sastoji u tome da se kroz algoritam umetanja
    prolazi vise puta. U prvom prolazu, umesto koraka 1 uzima se
46 neki korak h koji je manji od n (sto omogucuje razmenu
    udaljenih elemenata) i tako se dobija h-sortiran niz, tj. niz u
48 kome su elementi na rastojanju h sortirani, mada susedni
    elementi to ne moraju biti. U drugom prolazu kroz isti algoritam
    sprovodi se isti postupak ali za manji korak h. Sa prolazima se
50 nastavlja sve do koraka h = 1, u kome se dobija potpuno
    sortirani niz. Izbor pocetne vrednosti za h, i nacina njegovog
52 smanjivanja menja u nekim slucajevima brzinu algoritma, ali bilo
    koja vrednost ce rezultovati ispravnim sortiranjem, pod uslovom
    da je u poslednjoj iteraciji h imalo vrednost 1. */
54 void shell_sort(int a[], int n);

56 /* Merge sort: Funkcija sortira niz celih brojeva a[]
58 ucesljavanjem. Sortiranje se vrši od elementa na poziciji l do
60 onog na poziciji d. Na pocetku, da bi niz bio kompletno
    sortiran, l mora biti 0, a d je jednako poslednjem validnom
62 indeksu u nizu. Funkcija niz podeli na dve polovine, levu i
    desnu, koje zatim rekurzivno sortira. Od ova dva sortirana
64 podniza, sortiran niz se dobija ucesljavanjem, tj. istovremenim
    prolaskom kroz oba niza i izborom trenutnog manjeg elementa koji
66 se smesta u pomocni niz. Na kraju algoritma, sortirani elementi
    su u pomocnom nizu, koji se kopira u originalni niz. */

```

```

68 void merge_sort(int a[], int l, int d);

70 /* Quick sort: Funkcija sortira deo niza brojeva a izmedju pozicija
71    l i d. Njena ideja sortiranja je izbor jednog elementa niza, koji
72    se naziva pivot, i koji se dovodi na svoje mesto. Posle ovog
73    koraka, svi elementi levo od njega bice manji, a svi desno bice
74    veci od njega. Kako je pivot doveden na svoje mesto, da bi niz
75    bio kompletno sortiran, potrebno je sortirati elemente levo
76    (manje) od njega, i elemente desno (vece). Kako su dimenzije ova
77    dva podniza manje od dimenzije pocetnog niza koji je trebalo
78    sortirati, ovaj deo moze se uraditi rekurzivno. */
79 void quick_sort(int a[], int l, int d);
80
81 #endif

```

sort.c

```

1  #include "sort.h"

3  #define MAX 1000000

5  void selection_sort(int a[], int n)
6  {
7      int i, j;
8      int min;
9      int pom;

11     /* U svakoj iteraciji ove petlje pronalazi se najmanji element
12        medju elementima a[i], a[i+1],...,a[n-1], i postavlja se na
13        poziciju i, dok se element na poziciji i premesta na poziciju
14        min, na kojoj se nalazio najmanji od navedenih elemenata. */
15     for (i = 0; i < n - 1; i++) {
16         /* Unutrasnja petlja pronalazi poziciju min, na kojoj se nalazi
17            najmanji od elemenata a[i],...,a[n-1]. */
18         min = i;
19         for (j = i + 1; j < n; j++)
20             if (a[j] < a[min])
21                 min = j;

23         /* Zamena elemenata na pozicijama (i) i min. Ovo se radi samo
24            ako su (i) i min razliciti, inace je nepotrebno. */
25         if (min != i) {
26             pom = a[i];
27             a[i] = a[min];
28             a[min] = pom;
29         }
30     }
31 }

33 void insertion_sort(int a[], int n)
34 {

```

```

35     int i, j;

37     /* Na pocetku iteracije pretpostavlja se da je niz
        a[0],...,a[i-1] sortiran */
39     for (i = 1; i < n; i++) {
        /* U ovoj petlji se redom potiskuje element a[i] ulevo koliko
41         je potrebno, dok ne zauzme pravo mesto, tako da niz
        a[0],...,a[i] bude sortiran. Indeks j je trenutna pozicija na
43         kojoj se element koji se umece nalazi. Petlja se završava
        ili kada element dodje do levog kraja (j==0) ili kada se
45         naidje na element a[j-1] koji je manji od a[j]. */
        int temp = a[i];
47         for (j = i; j > 0 && temp < a[j - 1]; j--)
            a[j] = a[j - 1];
49         a[j] = temp;
    }
51 }

53 void bubble_sort(int a[], int n)
54 {
55     int i, j;
56     int ind;

57     for (i = n, ind = 1; i > 1 && ind; i--)
58     /* Poput "mehurica" potiskuje se najveći element među
        elementima od a[0] do a[i-1] na poziciju i-1 upoređujući
61         susedne elemente niza i potiskujući veći u desno */
        for (j = 0, ind = 0; j < i - 1; j++)
63         if (a[j] > a[j + 1]) {
            int temp = a[j];
65             a[j] = a[j + 1];
            a[j + 1] = temp;
67             /* Promenljiva ind registruje da je bilo premestanja. Samo
                u tom slučaju ima smisla ici na sledeću iteraciju, jer
69             ako nije bilo premestanja, znaci da su svi elementi već
                u dobrom poretku, pa nema potrebe prelaziti na kraći
71             prefiks niza. Algoritam bi bio korektan i bez ovoga.
                Sortiranje bi bilo ispravno, ali manje efikasno, jer bi
73             se cesto nepotrebno vrsila mnoga uporedjivanja, kada je
                već jasno da je sortiranje završeno. */
75             ind = 1;
        }
77 }

79 void shell_sort(int a[], int n)
80 {
81     int h = n / 2, i, j;
82     while (h > 0) {
83         /* Insertion sort sa korakom h */
84         for (i = h; i < n; i++) {
85             int temp = a[i];
            j = i;

```

```

87         while (j >= h && a[j - h] > temp) {
88             a[j] = a[j - h];
89             j -= h;
90         }
91         a[j] = temp;
92     }
93     h = h / 2;
94 }
95 }

97 void merge_sort(int a[], int l, int d)
98 {
99     int s;
100     static int b[MAX];          /* Pomocni niz */
101     int i, j, k;

102     /* Izlaz iz rekurzije */
103     if (l >= d)
104         return;

105     /* Odredjivanje sredisnjeg indeksa */
106     s = (l + d) / 2;

107     /* Rekurzivni pozivi */
108     merge_sort(a, l, s);
109     merge_sort(a, s + 1, d);

110     /* Inicijalizacija indeksa. Indeks i prolazi kroz levu polovinu
111        niza, dok indeks j prolazi kroz desnu polovinu niza. Indeks k
112        prolazi kroz pomocni niz b[] */
113     i = l;
114     j = s + 1;
115     k = 0;

116     /* "Ucesljavanje" koriscenjem pomocnog niza b[] */
117     while (i <= s && j <= d) {
118         if (a[i] < a[j])
119             b[k++] = a[i++];
120         else
121             b[k++] = a[j++];
122     }

123     /* U slucaju da se prethodna petlja zavrsla izlaskom promenljive
124        j iz dopustenog opsega u pomocni niz se prepisuje ostatak leve
125        polovine niza */
126     while (i <= s)
127         b[k++] = a[i++];

128     /* U slucaju da se prethodna petlja zavrsla izlaskom promenljive
129        i iz dopustenog opsega u pomocni niz se prepisuje ostatak desne
130        polovine niza */
131     while (j <= d)

```

```

139     b[k++] = a[j++];

141     /* Prepisuje se "ucesljani" niz u originalni niz */
    for (k = 0, i = 1; i <= d; i++, k++)
143         a[i] = b[k];
    }

145     /* Pomocna funkcija koja menja mesto i-tom i j-tom elementu niza a */
147 void swap(int a[], int i, int j)
    {
149         int tmp = a[i];
        a[i] = a[j];
151         a[j] = tmp;
    }

153 void quick_sort(int a[], int l, int d)
155 {
    int i, pivot_pozicija;

157     /* Izlaz iz rekurzije -- prazan niz */
    if (l >= d)
159         return;

161     /* Particionisanje niza. Svi elementi na pozicijama levo od
163        pivot_pozicija (izuzev same pozicije l) su strogo manji od
        pivota. Kada se pronadje neki element manji od pivota, uvecava
165        se promenljiva pivot_pozicija i na tu poziciju se premesta
        nadjeni element. Na kraju ce pivot_pozicija zaista biti
167        pozicija na koju treba smestiti pivot, jer ce svi elementi levo
        od te pozicije biti manji a desno biti veci ili jednaki od
169        pivota. */
    pivot_pozicija = l;
171    for (i = l + 1; i <= d; i++)
        if (a[i] < a[l])
173        swap(a, ++pivot_pozicija, i);

175    /* Postavljanje pivota na svoje mesto */
    swap(a, l, pivot_pozicija);

177    /* Rekurzivno sortiranje elementa manjih od pivota */
    quick_sort(a, l, pivot_pozicija - 1);
179    /* Rekurzivno sortiranje elementa vecih od pivota */
    quick_sort(a, pivot_pozicija + 1, d);
181 }

```

main.c

```

#include <stdio.h>
2  #include <stdlib.h>
#include <time.h>
4  #include "sort.h"

```

```

6  /* Maksimalna duzina niza */
   #define MAX 1000000
8
10 int main(int argc, char *argv[])
   {
12     /******
        tip_sortiranja == 0 => selectionsort, (podrazumevano)
        tip_sortiranja == 1 => insertionsort, -i opcija komandne linije
14     tip_sortiranja == 2 => bubblesort,    -b opcija komandne linije
        tip_sortiranja == 3 => shellsort,    -s opcija komandne linije
16     tip_sortiranja == 4 => mergesort,     -m opcija komandne linije
        tip_sortiranja == 5 => quicksort,    -q opcija komandne linije
18     *****/
        int tip_sortiranja = 0;
20     /******
        tip_niza == 0 => slucajno generisani nizovi, (podrazumevano)
        tip_niza == 1 => rastuce sortirani nizovi,  -r opcija
        tip_niza == 2 => opadajuce soritrani nizovi, -o opcija
24     *****/
        int tip_niza = 0;
26
        /* Dimenzija niza koji se sortira */
28     int dimenzija;
        int i;
30     int niz[MAX];

32     /* Provera argumenata komandne linije */
        if (argc < 2) {
34         fprintf(stderr, "Greska: Program zahteva bar 2 ");
            fprintf(stderr, "argumenta komandne linije!\n");
36         exit(EXIT_FAILURE);
        }
38
        /* Ocitavaju se opcije i argumenati komandne linije */
40     for (i = 1; i < argc; i++) {
        /* Ako je u pitanju opcija... */
42         if (argv[i][0] == '-') {
            switch (argv[i][1]) {
44                 case 'i':
                    tip_sortiranja = 1;
46                     break;
                    case 'b':
48                     tip_sortiranja = 2;
                        break;
                    case 's':
50                     tip_sortiranja = 3;
                        break;
                    case 'm':
52                     tip_sortiranja = 4;
                        break;
                    case 'q':
54                     tip_sortiranja = 5;
                        break;
                    case 'r':
56                     tip_sortiranja = 1;
                        break;
                    case 'o':
                    tip_sortiranja = 2;
                        break;
            }
        }
    }
}

```

```

58     tip_sortiranja = 5;
        break;
60     case 'r':
        tip_niza = 1;
        break;
62     case 'o':
        tip_niza = 2;
64     break;
        default:
66         fprintf(stderr, "Greska: Pogresna opcija -%c\n",
            argv[i][1]);
68         exit(EXIT_SUCCESS);
        break;
70     }
    }
72     /* Ako je u pitanju argument, onda je to duzina niza koji treba
        da se sortira */
74     else {
        dimenzija = atoi(argv[i]);
76         if (dimenzija <= 0 || dimenzija > MAX) {
            fprintf(stderr, "Greska: Dimenzija niza neodgovarajuca!\n");
78             exit(EXIT_FAILURE);
        }
80     }
}

82
/* Elementi niza se odredjuju slucajno, ali vodeci racuna o tipu
84 niza dobijenom iz komandne linije. srand() funkcija
    obezbedjuje novi seed za pozivanje rand funkcije, i kako
86 generisani niz ne bi uvek bio isti seed je postavljen na
    tekuce vreme u sekundama od Nove godine 1970. rand()%100 daje
88 brojeve izmedju 0 i 99 */
srand(time(NULL));
90 if (tip_niza == 0)
    for (i = 0; i < dimenzija; i++)
92         niz[i] = rand();
    else if (tip_niza == 1)
94         for (i = 0; i < dimenzija; i++)
            niz[i] = i == 0 ? rand() % 100 : niz[i - 1] + rand() % 100;
96     else
        for (i = 0; i < dimenzija; i++)
98             niz[i] = i == 0 ? rand() % 100 : niz[i - 1] - rand() % 100;

100 /* Ispisuju se elemenati niza */
    /******
102     Ovaj deo je iskomentarisan jer sledeci ispis ne treba da se nadje
        na standardnom izlazu. Njegova svrha je samo bila provera da li
104     je niz generisan u skladu sa opcijama komandne linije.

106     printf("Niz koji sortiramo je:\n");
        for (i = 0; i < dimenzija; i++)
108             printf("%d\n", niz[i]);

```

```

110  *****/
111
112  /* Sortira se niz na odgovarajuci nacin */
113  if (tip_sortiranja == 0)
114      selection_sort(niz, dimenzija);
115  else if (tip_sortiranja == 1)
116      insertion_sort(niz, dimenzija);
117  else if (tip_sortiranja == 2)
118      bubble_sort(niz, dimenzija);
119  else if (tip_sortiranja == 3)
120      shell_sort(niz, dimenzija);
121  else if (tip_sortiranja == 4)
122      merge_sort(niz, 0, dimenzija - 1);
123  else
124      quick_sort(niz, 0, dimenzija - 1);
125
126  /* Ispisuju se elementi niza */
127  /******
128   Ovaj deo je iskomentarisan jer vreme potrebno za njegovo
129   izvršavanje ne bi trebalo da bude ukljuceno u vreme izmereno
130   programom time. Takodje, kako je svrha ovog programa da prikaze
131   vremena razlicitih algoritama sortiranja, dimenzije nizova ce
132   biti, verovatno, ogromne, pa nema smisla imati na izlazu nizove
133   od toliko elemenata. Ovaj deo je koriscen u razvoju programa
134   zarad testiranja korektnosti.
135
136   printf("Sortiran niz je:\n");
137   for (i = 0; i < dimenzija; i++)
138       printf("%d\n", niz[i]);
139  *****/
140  exit(EXIT_SUCCESS);
141  }

```

Rešenje 1.13

```

1  #include <stdio.h>
2  #include <string.h>
3
4  #define MAX_DIM 128
5
6  /* Funkcija za sortiranje niza karaktera */
7  void selectionSort(char s[])
8  {
9      int i, j, min;
10     char pom;
11     for (i = 0; s[i] != '\0'; i++) {
12         min = i;
13         for (j = i + 1; s[j] != '\0'; j++)
14             if (s[j] < s[min])
15                 min = j;
16         pom = s[i];
17         s[i] = s[min];
18         s[min] = pom;
19     }
20 }

```



```

16     if (min != i) {
17         pom = s[i];
18         s[i] = s[min];
19         s[min] = pom;
20     }
21 }
22 }

24 /* Funkcija vraca 1 ako su argumenti anagrami, a 0 inace. */
25 int anagrami(char s[], char t[])
26 {
27     int i;
28
29     /* Ako dve niske imaju razlicit broj karaktera onda one nisu
30        anagrami */
31     if (strlen(s) != strlen(t))
32         return 0;
33
34     /* Sortiraju se karakteri */
35     selectionSort(s);
36     selectionSort(t);
37
38     /* Dve sortirane niske su anagrami ako i samo ako su jednake */
39     for (i = 0; s[i] != '\0'; i++)
40         if (s[i] != t[i])
41             return 0;
42     return 1;
43 }
44
45 int main()
46 {
47     char s[MAX_DIM], t[MAX_DIM];
48
49     /* Ucitavaju se niske sa ulaza */
50     printf("Unesite prvu nisku: ");
51     scanf("%s", s);
52     printf("Unesite drugu nisku: ");
53     scanf("%s", t);
54
55     /* Poziv funkcije */
56     if (anagrami(s, t))
57         printf("jesu\n");
58     else
59         printf("nisu\n");
60
61     return 0;
62 }

```

Rešenje 1.14

NAPOMENA: Rešenje koristi biblioteku za sortiranje celih brojeva iz zadatka 1.12.

```
1 #include <stdio.h>
2 #include "sort.h"
3
4 #define MAX 256
5
6 /* Funkcija koja pronalazi najmanje rastojanje izmedju dva broja u
7    sortiranom nizu celih brojeva */
8 int najmanje_rastojanje(int a[], int n)
9 {
10     int i, min;
11     /* Postavlja se najmanje rastojanje na razliku prvog i drugog
12        elementa niza */
13     min = a[1] - a[0];
14     /* Za sve ostale susedne elemente proverava se njigova razlika */
15     for (i = 2; i < n; i++)
16         if (a[i] - a[i - 1] < min)
17             min = a[i] - a[i - 1];
18     return min;
19 }
20
21 int main()
22 {
23     int i, a[MAX];
24
25     /* Ucitavaju se elementi niza sve do kraja ulaza */
26     i = 0;
27     while (scanf("%d", &a[i]) != EOF)
28         i++;
29
30     /* Za sortiranje niza moze se koristiti bilo koja od funkcija
31        sortiranja iz sort.h. Ilustracije radi, u ovom zadatku koristi
32        se selection sort. */
33     selection_sort(a, i);
34
35     /* Ispisuje se rezultat */
36     printf("%d\n", najmanje_rastojanje(a, i));
37
38     return 0;
39 }
```

Rešenje 1.15

NAPOMENA: Rešenje koristi biblioteku za sortiranje celih brojeva iz zadatka 1.12.

```
1 #include <stdio.h>
2 #include "sort.h"
3
4 #define MAX_DIM 256
```

```

5
7  /* Funkcija za odredjivanje onog elementa sortiranog niza koji se
   najviše puta pojavio u tom nizu */
9  int najvise_puta(int a[], int n)
10 {
11     int i, j, br_pojava, i_max_pojava = -1, max_br_pojava = -1;
12     /* Za i-ti element izracunava se koliko puta se pojavio u nizu */
13     for (i = 0; i < n; i = j) {
14         br_pojava = 1;
15         for (j = i + 1; j < n && a[i] == a[j]; j++)
16             br_pojava++;
17         /* Ispituje se da li se do tog trenutka i-ti element pojavio
18            najviše puta u nizu */
19         if (br_pojava > max_br_pojava) {
20             max_br_pojava = br_pojava;
21             i_max_pojava = i;
22         }
23     }
24     /* Vraca se element koji se najviše puta pojavio u nizu */
25     return a[i_max_pojava];
26 }
27
28 int main()
29 {
30     int a[MAX_DIM], i;
31
32     /* Ucitavaju se elemenati niza sve do kraja ulaza */
33     i = 0;
34     while (scanf("%d", &a[i]) != EOF)
35         i++;
36
37     /* Za sortiranje niza moze se koristiti bilo koja od funkcija
38        sortiranja iz sort.h. Ilustracije radi, u ovom zadatku koristi
39        se merge sort. */
40     merge_sort(a, 0, i - 1);
41
42     /* Odredjuje se broj koji se najviše puta pojavio u nizu */
43     printf("%d\n", najvise_puta(a, i));
44
45     return 0;
46 }

```

Rešenje 1.16

NAPOMENA: Rešenje koristi biblioteku za sortiranje celih brojeva iz zadatka [1.12](#).

```

1  #include <stdio.h>
2  #include "sort.h"
3
4  #define MAX_DIM 256
5

```

```

7      /* Funkcija za binarnu pretragu niza vraca 1 ako se element x
      nalazi u nizu, a 0 inace. Pretpostavlja se da je niz sortiran u
      rastucem poretku */
9      int binarna_pretraga(int a[], int n, int x)
      {
11         int levi = 0, desni = n - 1, srednji;

13         while (levi <= desni) {
            srednji = (levi + desni) / 2;
15             if (a[srednji] == x)
                return 1;
17             else if (a[srednji] > x)
                desni = srednji - 1;
19             else if (a[srednji] < x)
                levi = srednji + 1;
21         }
            return 0;
23     }

25     int main()
    {
27         int a[MAX_DIM], n = 0, zbir, i;

29         /* Ucitava se trazeni zbir */
        printf("Unesite trazeni zbir: ");
31         scanf("%d", &zbir);

33         /* Ucitavaju se elementi niza sve do kraja ulaza */
        i = 0;
35         printf("Unesite elemente niza: ");
        while (scanf("%d", &a[i]) != EOF)
37             i++;
        n = i;
39

41         /* Za sortiranje niza moze se koristiti bilo koja od funkcija
        sortiranja iz sort.h. Ilustracije radi, u ovom zadatku koristi
        se quick sort. */
43         quick_sort(a, 0, n - 1);

45         for (i = 0; i < n; i++)
            /* Za i-ti element niza binarno se pretrazuje da li se u
            ostatku niza nalazi element koji sabran sa njim ima ucitanu
            vrednost zbira */
49             if (binarna_pretraga(a + i + 1, n - i - 1, zbir - a[i])) {
                printf("da\n");
51                 return 0;
            }
53         printf("ne\n");

55         return 0;
    }

```

Rešenje 1.17

```
1  #include <stdio.h>
2  #define MAX_DIM 256
3
4  /* Funkcija objedinjuje nizove niz1 i niz2 dimenzija dim1 i dim2, a
5     rezultat cuva u nizu niz3 za koji je rezervisano dim3 elemenata */
6  int merge(int *niz1, int dim1, int *niz2, int dim2, int *niz3,
7            int dim3)
8  {
9      int i = 0, j = 0, k = 0;
10     /* U slucaju da je dimenzija treceg niza manja od neophodne,
11        funkcija vraca -1 */
12     if (dim3 < dim1 + dim2)
13         return -1;
14
15     /* Vrsi se ucesljavanje nizova sve dok se ne dodje do kraja
16        jednog od njih */
17     while (i < dim1 && j < dim2) {
18         if (niz1[i] < niz2[j])
19             niz3[k++] = niz1[i++];
20         else
21             niz3[k++] = niz2[j++];
22     }
23     /* Ostatak prvog niza prepisuje se u treci */
24     while (i < dim1)
25         niz3[k++] = niz1[i++];
26
27     /* Ostatak drugog niza prepisuje se u treci */
28     while (j < dim2)
29         niz3[k++] = niz2[j++];
30     return dim1 + dim2;
31 }
32
33 int main()
34 {
35     int niz1[MAX_DIM], niz2[MAX_DIM], niz3[2 * MAX_DIM];
36     int i = 0, j = 0, k, dim3;
37
38     /* Ucitavaju se nizovi sa ulaza sve dok se ne unese nula.
39        Pretpostavka je da na ulazu nema vise od MAX_DIM elemenata */
40     printf("Unesite elemente prvog niza: ");
41     while (1) {
42         scanf("%d", &niz1[i]);
43         if (niz1[i] == 0)
44             break;
45         i++;
46     }
47     printf("Unesite elemente drugog niza: ");
48     while (1) {
49         scanf("%d", &niz2[j]);
50         if (niz2[j] == 0)
```

```

        break;
52     j++;
    }

54     /* Poziv trazene funkcije */
56     dim3 = merge(niz1, i, niz2, j, niz3, 2 * MAX_DIM);

58     /* Ispis niza */
    for (k = 0; k < dim3; k++)
60         printf("%d ", niz3[k]);
    printf("\n");

62     return 0;
64 }

```

Rešenje 1.18

```

#include <stdio.h>
2  #include <stdlib.h>
  #include <string.h>

4
int main(int argc, char *argv[])
6  {
    FILE *fin1 = NULL, *fin2 = NULL;
    FILE *fout = NULL;
    char ime1[11], ime2[11];
10   char prezime1[16], prezime2[16];
    int kraj1 = 0, kraj2 = 0;

12
    /* Ako nema dovoljno argumenata komandne linije */
14   if (argc < 3) {
        fprintf(stderr,
16             "Greska: Program se poziva sa %s datoteka1 datoteka2\n",
                argv[0]);
18     exit(EXIT_FAILURE);
    }

20
    /* Otvara se datoteka zadata prvim argumentom komandne linije */
22   fin1 = fopen(argv[1], "r");
    if (fin1 == NULL) {
24       fprintf(stderr, "Greska: Neuspesno otvaranje datoteke %s\n",
                argv[1]);
26       exit(EXIT_FAILURE);
    }

28
    /* Otvara se datoteka zadata drugim argumentom komandne linije */
30   fin2 = fopen(argv[2], "r");
    if (fin2 == NULL) {
32       fprintf(stderr, "Greska: Neuspesno otvaranje datoteke %s\n",
                argv[2]);
34       exit(EXIT_FAILURE);
    }

```

```

36     }
37
38     /* Otvara se datoteka za upis rezultata */
39     fout = fopen("ceo-tok.txt", "w");
40     if (fout == NULL) {
41         fprintf(stderr,
42             "Greska: Neuspesno otvaranje datoteke ceo-tok.txt\n");
43         exit(EXIT_FAILURE);
44     }
45
46     /* Cita se prvi student iz prve datoteke */
47     if (fscanf(fin1, "%s%s", ime1, prezime1) == EOF)
48         kraj1 = 1;
49
50     /* Cita se prvi student iz druge datoteke */
51     if (fscanf(fin2, "%s%s", ime2, prezime2) == EOF)
52         kraj2 = 1;
53
54     /* Sve dok nije dostignut kraj neke datoteke */
55     while (!kraj1 && !kraj2) {
56         int tmp = strcmp(ime1, ime2);
57         if (tmp < 0 || (tmp == 0 && strcmp(prezime1, prezime2) < 0)) {
58             /* Ime i prezime iz prve datoteke je leksikografski ranije, i
59              biva upisano u izlaznu datoteku */
60             fprintf(fout, "%s %s\n", ime1, prezime1);
61             /* Cita se naredni student iz prve datoteke */
62             if (fscanf(fin1, "%s%s", ime1, prezime1) == EOF)
63                 kraj1 = 1;
64         } else {
65             /* Ime i prezime iz druge datoteke je leksikografski ranije,
66              i biva upisano u izlaznu datoteku */
67             fprintf(fout, "%s %s\n", ime2, prezime2);
68             /* Cita se naredni student iz druge datoteke */
69             if (fscanf(fin2, "%s%s", ime2, prezime2) == EOF)
70                 kraj2 = 1;
71         }
72     }
73
74     /* Ako se iz prethodne petlje izaslo zato sto je dostignut kraj
75     druge datoteke, onda ima jos studenata u prvoj datoteci, koje
76     treba prepisati u izlaznu, redom, jer su vec sortirani po
77     imenu. */
78     while (!kraj1) {
79         fprintf(fout, "%s %s\n", ime1, prezime1);
80         if (fscanf(fin1, "%s%s", ime1, prezime1) == EOF)
81             kraj1 = 1;
82     }
83
84     /* Ako se iz prve petlje izaslo zato sto je dostignut kraj prve
85     datoteke, onda ima jos studenata u drugoj datoteci, koje treba
86     prepisati u izlaznu, redom, jer su vec sortirani po imenu. */
87     while (!kraj2) {

```

```

    fprintf(fout, "%s %s\n", ime2, prezime2);
88     if (fscanf(fin2, "%s%s", ime2, prezime2) == EOF)
        kraj2 = 1;
90 }

92 /* Zatvaraju se datoteke */
    fclose(fin1);
94     fclose(fin2);
    fclose(fout);

96     exit(EXIT_SUCCESS);
98 }

```

Rešenje 1.19

```

1  #include <stdio.h>
   #include <string.h>
3  #include <math.h>
   #include <stdlib.h>
5
   #define MAX_BR_TACAKA 128
7
   /* Struktura koja reprezentuje koordinate tacke */
9  typedef struct Tacka {
    int x;
11     int y;
    } Tacka;
13
   /* Funkcija racuna rastojanje zadate tacke od koordinatnog pocetka */
15 float rastojanje(Tacka A)
    {
17     return sqrt(A.x * A.x + A.y * A.y);
    }
19
   /* Funkcija koja sortira niz tacaka po rastojanju od koordinatnog
21     pocetka */
void sortiraj_po_rastojanju(Tacka t[], int n)
23 {
    int min, i, j;
25     Tacka tmp;

27     for (i = 0; i < n - 1; i++) {
        min = i;
29         for (j = i + 1; j < n; j++) {
            if (rastojanje(t[j]) < rastojanje(t[min])) {
31                 min = j;
            }
33         }
        if (min != i) {
35             tmp = t[i];
            t[i] = t[min];

```



```

37         t[min] = tmp;
38     }
39 }
40 }
41
42 /* Funkcija koja sortira niz tacaka po vrednosti x koordinate */
43 void sortiraj_po_x(Tacka t[], int n)
44 {
45     int min, i, j;
46     Tacka tmp;
47
48     for (i = 0; i < n - 1; i++) {
49         min = i;
50         for (j = i + 1; j < n; j++) {
51             if (abs(t[j].x) < abs(t[min].x)) {
52                 min = j;
53             }
54         }
55         if (min != i) {
56             tmp = t[i];
57             t[i] = t[min];
58             t[min] = tmp;
59         }
60     }
61 }
62
63 /* Funkcija koja sortira niz tacaka po vrednosti y koordinate */
64 void sortiraj_po_y(Tacka t[], int n)
65 {
66     int min, i, j;
67     Tacka tmp;
68
69     for (i = 0; i < n - 1; i++) {
70         min = i;
71         for (j = i + 1; j < n; j++) {
72             if (abs(t[j].y) < abs(t[min].y)) {
73                 min = j;
74             }
75         }
76         if (min != i) {
77             tmp = t[i];
78             t[i] = t[min];
79             t[min] = tmp;
80         }
81     }
82 }
83
84 int main(int argc, char *argv[])
85 {
86     FILE *ulaz;
87     FILE *izlaz;
88     Tacka tacke[MAX_BR_TACAKA];

```

```

89     int i, n;

91     /* Proverava se broj argumenata komandne linije. Ocekuje se ime
92        izvrsnog programa, opcija, ime ulazne datoteke i ime izlazne
93        datoteke, tj. 4 argumenta */
94     if (argc != 4) {
95         fprintf(stderr,
96             "Greska: Program se poziva sa %s opcija ulaz izlaz\n",
97             argv[0]);
98         exit(EXIT_FAILURE);
99     }

101    /* Otvara se datoteka u kojoj su zadate tacke */
102    ulaz = fopen(argv[2], "r");
103    if (ulaz == NULL) {
104        fprintf(stderr, "Greska: Neuspesno otvaranje datoteke %s!\n",
105            argv[2]);
106        exit(EXIT_FAILURE);
107    }

109    /* Otvara se datoteka u koju treba upisati rezultat */
110    izlaz = fopen(argv[3], "w");
111    if (izlaz == NULL) {
112        fprintf(stderr, "Greska: Neuspesno otvaranje datoteke %s!\n",
113            argv[3]);
114        exit(EXIT_FAILURE);
115    }

117    /* Sve dok se ne stigne do kraja ulazne datoteke, ucitavaju se
118       koordinate tacaka i smestaju na odgovarajuce pozicije
119       odredjene brojacem i. */
120    i = 0;
121    while (fscanf(ulaz, "%d %d", &tacke[i].x, &tacke[i].y) != EOF) {
122        i++;
123    }

125    /* Ukupan broj procitanih tacaka */
126    n = i;

127

128    /* Analizira se prosledjena opcija. Moguce vrednosti za argv[1]
129       su "-x" ili "-y" ili "-o", pa je argv[1][0] sigurno crtica
130       (karakter -), a karakter argv[1][1] odredjuje kriterijum
131       sortiranja */
132    switch (argv[1][1]) {
133    case 'x':
134        /* Sortira se po vrednosti x koordinate */
135        sortiraj_po_x(tacke, n);
136        break;
137    case 'y':
138        /* Sortira se po vrednosti y koordinate */
139        sortiraj_po_y(tacke, n);
140        break;

```

```

141     case 'o':
142         /* Sortira se po udaljenosti od koordinatnog pocetka */
143         sortiraj_po_rastojanju(tacke, n);
144         break;
145     }

147     /* Niz se upisuje u izlaznu datoteku */
148     for (i = 0; i < n; i++) {
149         fprintf(izlaz, "%d %d\n", tacke[i].x, tacke[i].y);
150     }

151     /* Zatvaraju se otvorene datoteke */
152     fclose(ulaz);
153     fclose(izlaz);

154     exit(EXIT_SUCCESS);
155 }
156
157 }

```

Rešenje 1.20

```

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define MAX 1000
#define MAX_DUZINA 16

/* Struktura koja reprezentuje jednog gradjanina */
typedef struct gr {
    char ime[MAX_DUZINA];
    char prezime[MAX_DUZINA];
} Gradjanin;

/* Funkcija sortira niz gradjana rastuce po imenima */
void sort_ime(Gradjanin a[], int n)
{
    int i, j, min;
    Gradjanin pom;

    for (i = 0; i < n - 1; i++) {
        /* Unutrasnja petlja pronalazi poziciju min, na kojoj se nalazi
        najmanji od elemenata a[i].ime, ..., a[n-1].ime. */
        min = i;
        for (j = i + 1; j < n; j++)
            if (strcmp(a[j].ime, a[min].ime) < 0)
                min = j;
        /* Zamenjena elementa na pozicijama (i) i min. Ovo se radi samo
        ako su (i) i min razliciti, inace je nepotrebno. */
        if (min != i) {
            pom = a[i];
            a[i] = a[min];
            a[min] = pom;
        }
    }
}

```

```

32     a[min] = pom;
33     }
34 }
35 }
36
37 /* Funkcija sortira niz gradjana rastuce po prezimenima */
38 void sort_prezime(Gradjanin a[], int n)
39 {
40     int i, j, min;
41     Gradjanin pom;
42
43     for (i = 0; i < n - 1; i++) {
44         /* Unutrasnja petlja pronalazi poziciju min, na kojoj se nalazi
45            najmanji od elemenata a[i].prezime,...,a[n-1].prezime. */
46         min = i;
47         for (j = i + 1; j < n; j++)
48             if (strcmp(a[j].prezime, a[min].prezime) < 0)
49                 min = j;
50         /* Zamena elemenata na pozicijama (i) i min. Ovo se radi samo
51            ako su (i) i min razliciti, inace je nepotrebno. */
52         if (min != i) {
53             pom = a[i];
54             a[i] = a[min];
55             a[min] = pom;
56         }
57     }
58 }
59
60 /* Pretraga niza gradjana */
61 int linearna_pretraga(Gradjanin a[], int n, Gradjanin * x)
62 {
63     int i;
64     for (i = 0; i < n; i++)
65         if (strcmp(a[i].ime, x->ime) == 0
66             && strcmp(a[i].prezime, x->prezime) == 0)
67             return i;
68     return -1;
69 }
70
71
72 int main()
73 {
74     Gradjanin spisak1[MAX], spisak2[MAX];
75     int isti_rbr = 0;
76     int i, n;
77     FILE *fp = NULL;
78
79     /* Otvara se datoteka */
80     if ((fp = fopen("biracki-spisak.txt", "r")) == NULL) {
81         fprintf(stderr,
82             "Greska: Neuspesno otvaranje datoteke za citanje.\n");
83         exit(EXIT_FAILURE);

```

```

84     }

86     /* Cita se sadrzaj */
87     for (i = 0;
88         fscanf(fp, "%s %s", spisak1[i].ime,
89                spisak1[i].prezime) != EOF; i++)
90         spisak2[i] = spisak1[i];
91     n = i;

92     /* Zatvara se datoteka */
93     fclose(fp);

94     sort_ime(spisak1, n);

95
96     /*****
97     Ovaj deo je iskomentaran jer se u zadatku ne trazi ispis
98     sortiranih nizova. Koristi se samo u fazi testiranja programa.
99
100     printf("Biracki spisak [uredjen prema imenima]:\n");
101     for(i=0; i<n; i++)
102         printf(" %d. %s %s\n",i,spisak1[i].ime, spisak1[i].prezime);
103     *****/

104     sort_prezime(spisak2, n);

105
106     /*****
107     Ovaj deo je iskomentaran jer se u zadatku ne trazi ispis
108     sortiranih nizova. Koristi se samo u fazi testiranja programa.
109
110     printf("Biracki spisak [uredjen prema prezimenima]:\n");
111     for(i=0; i<n; i++)
112         printf(" %d. %s %s\n",i,spisak2[i].ime, spisak2[i].prezime);
113     *****/

114     /* Linearno se pretrazuju nizovi */
115     for (i = 0; i < n; i++)
116         if (i == linearna_pretraga(spisak2, n, &spisak1[i]))
117             isti_rbr++;

118
119     /* Alternativno (efikasnije) resenje */
120     /*****
121     for(i=0; i<n ;i++)
122         if( strcmp(spisak2[i].ime, spisak1[i].ime) == 0 &&
123            strcmp(spisak1[i].prezime, spisak2[i].prezime)==0)
124             isti_rbr++;
125     *****/

126
127     /* Ispisuje se rezultat */
128     printf("%d\n", isti_rbr);

129
130     exit(EXIT_SUCCESS);
131 }

```

Rešenje 1.22

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <ctype.h>
5
6 #define MAX_BR_RECII 128
7 #define MAX_DUZINA_RECII 32
8
9 /* Funkcija koja izracunava broj suglasnika u reci */
10 int broj_suglasnika(char s[])
11 {
12     char c;
13     int i;
14     int suglasnici = 0;
15     /* Prolaz karakter po karakter kroz zadatu nisku */
16     for (i = 0; s[i]; i++) {
17         /* Ako je u pitanju slovo, konvertuje se u veliko da bi bio
18            pokriven slucaj i malih i velikih suglasnika. */
19         if (isalpha(s[i])) {
20             c = toupper(s[i]);
21             /* Ukoliko slovo nije samoglasnik uvecava se brojac. */
22             if (c != 'A' && c != 'E' && c != 'I' && c != 'O' && c != 'U')
23                 suglasnici++;
24         }
25     }
26     /* Vraca se izracunata vrednost */
27     return suglasnici;
28 }
29
30 /* Funkcija koja sortira reci po zadatom kriterijumu. Informacija o
31    duzini reci se mora proslediti zbog pravilnog upravljanja
32    memorijom */
33 void sortiraj_reci(char reci[][MAX_DUZINA_RECII], int n)
34 {
35     int min, i, j, broj_suglasnika_j, broj_suglasnika_min,
36         duzina_j, duzina_min;
37     char tmp[MAX_DUZINA_RECII];
38     for (i = 0; i < n - 1; i++) {
39         min = i;
40         for (j = i; j < n; j++) {
41             /* Prvo se uporedjuje broj suglasnika */
42             broj_suglasnika_j = broj_suglasnika(reci[j]);
43             broj_suglasnika_min = broj_suglasnika(reci[min]);
44             if (broj_suglasnika_j < broj_suglasnika_min)
45                 min = j;
46             else if (broj_suglasnika_j == broj_suglasnika_min) {
47                 /* Zatim, recima koje imaju isti broj suglasnika uporedjuju
48                    se duzine */
49                 duzina_j = strlen(reci[j]);
50                 duzina_min = strlen(reci[min]);
51             }
52         }
53         /* Zamena mesta */
54         strcpy(tmp, reci[i]);
55         strcpy(reci[i], reci[min]);
56         strcpy(reci[min], tmp);
57     }
58 }
```

```

51         if (duzina_j < duzina_min)
52             min = j;
53         else {
54             /* Ako reci imaju i isti broj suglasnika i iste duzine,
55                uporedjuju se leksikografski */
56             if (duzina_j == duzina_min
57                 && strcmp(reci[j], reci[min]) < 0)
58                 min = j;
59         }
60     }
61 }
62
63 if (min != i) {
64     strcpy(tmp, reci[min]);
65     strcpy(reci[min], reci[i]);
66     strcpy(reci[i], tmp);
67 }
68 }
69 }
70
71 int main()
72 {
73     FILE *ulaz;
74     int i = 0, n;
75
76     /* Niz u koji ce biti smestane reci. Prvi broj oznacava broj
77        reci, a drugi maksimalnu duzinu pojedinačne reci */
78     char reci[MAX_BR_RECII][MAX_DUZINA_RECII];
79
80     /* Otvara se datoteka niske.txt za citanje */
81     ulaz = fopen("niske.txt", "r");
82     if (ulaz == NULL) {
83         fprintf(stderr,
84             "Greska: Neuspesno otvaranje datoteke niske.txt!\n");
85         exit(EXIT_FAILURE);
86     }
87
88     /* Sve dok se moze procitati sledeca rec */
89     while (fscanf(ulaz, "%s", reci[i]) != EOF) {
90         /* Proverava se da li ucitan maksimalan broj reci, i ako jeste,
91            prekida se učitavanje */
92         if (i == MAX_BR_RECII)
93             break;
94         /* Priprema brojaca za narednu iteraciju */
95         i++;
96     }
97
98     /* n je duzina niza reci i predstavlja poslednju vrednost
99        koriscenog brojaca */
100     n = i;
101     /* Poziva se funkcija za sortiranje reci */
102     sortiraj_reci(reci, n);

```

```

103     /* Ispis sortiranog niza reci */
105     for (i = 0; i < n; i++) {
106         printf("%s ", reci[i]);
107     }
108     printf("\n");
109
110     /* Zatvara se datoteka */
111     fclose(ulaz);
112
113     exit(EXIT_SUCCESS);
114 }

```

Rešenje 1.23

```

#include <stdio.h>
2 #include <stdlib.h>
#include <string.h>
4
#define MAX_ARTIKALA 100000
6
/* Struktura koja predstavlja jedan artikal */
8 typedef struct art {
    long kod;
10    char naziv[20];
    char proizvodjac[20];
12    float cena;
} Artikal;
14
/* Funkcija koja u nizu artikala binarnom pretragom nalazi onaj sa
16    traženim bar kodom */
17 int binarna_pretraga(Artikal a[], int n, long x)
18 {
    int levi = 0;
20    int desni = n - 1;
21
22    /* Dokle god je indeks levi levo od indeksa desni */
    while (levi <= desni) {
23        /* Racuna se sredisnji indeks */
24        int srednji = (levi + desni) / 2;
25        /* Ako je sredisnji element veci od traženog, tada se traženi
26            mora nalaziti u levoj polovini niza */
27        if (x < a[srednji].kod)
28            desni = srednji - 1;
29        /* Ako je sredisnji element manji od traženog, tada se traženi
30            mora nalaziti u desnoj polovini niza */
31        else if (x > a[srednji].kod)
32            levi = srednji + 1;
33        else
34            /* Ako je sredisnji element jednak traženom, tada je artikal
35                sa bar kodom x pronadjen na poziciji srednji */

```



```

38     return srednji;
39 }
40 /* Ako nije pronadjen artikal za trazanim bar kodom, vraca se -1 */
41 return -1;
42 }
43
44 /* Funkcija koja sortira niz artikala po bar kodovima rastuce */
45 void selection_sort(Artikal a[], int n)
46 {
47     int i, j;
48     int min;
49     Artikal pom;
50
51     for (i = 0; i < n - 1; i++) {
52         min = i;
53         for (j = i + 1; j < n; j++)
54             if (a[j].kod < a[min].kod)
55                 min = j;
56         if (min != i) {
57             pom = a[i];
58             a[i] = a[min];
59             a[min] = pom;
60         }
61     }
62 }
63
64 int main()
65 {
66     Artikal asortiman[MAX_ARTIKALA];
67     long kod;
68     int i, n;
69     float racun;
70     FILE *fp = NULL;
71
72     /* Otvara se datoteka */
73     if ((fp = fopen("artikli.txt", "r")) == NULL) {
74         fprintf(stderr,
75             "Greska: Neuspesno otvaranje datoteke artikli.txt.\n");
76         exit(EXIT_FAILURE);
77     }
78
79     /* Ucitavaju se artikali */
80     i = 0;
81     while (fscanf(fp, "%ld %s %s %f", &asortiman[i].kod,
82         asortiman[i].naziv, asortiman[i].proizvodjac,
83         &asortiman[i].cena) == 4)
84         i++;
85
86     /* Zatvara se datoteka */
87     fclose(fp);
88
89     n = i;

```

```

90  /* Sortira se celokupan asortiman prodavnice prema kodovima jer
91     ce pri kucanju racuna prodavac unositi kod artikla. Prilikom
92     kucanja svakog racuna pretrazuje se asortiman, da bi se
93     utvrdila cena artikla. Kucanje racuna obuhvata vise pretraga
94     asortimana i cilj je da ta operacija bude sto efikasnija.
95     Zato se koristi algoritam binarne pretrage prilikom
96     pretrazivanja po kodu artikla. Iz tog razloga, potrebno je da
97     asortiman bude sortirani po kodovima i to ce biti uradjeno
98     primenom selection sort algoritma. Sortiranje se vrši samo
99     jednom na pocetku, ali se zato posle artikli mogu brzo
100    pretrazivati prilikom kucanja proizvoljno puno racuna. Vreme
101    koje se utrosi na sortiranje na pocetku izvršavanja programa,
102    kasnije se isplati jer se za brojna trazjenja artikla umesto
103    linearne moze koristiti efikasnija binarna pretraga. */
104    selection_sort(asortiman, n);

106    /* Ispis stanja u prodavnici */
107    printf
108    ( "Asortiman:\nKOD          Naziv artikla      Ime
109      proizvodjaca      Cena\n");
110    for (i = 0; i < n; i++)
111        printf("%10ld %20s %20s %12.2f\n", asortiman[i].kod,
112            asortiman[i].naziv, asortiman[i].proizvodjac,
113            asortiman[i].cena);

114    kod = 0;
115    while (1) {
116        printf("-----\n");
117        printf("- Za kraj za kraj rada kase, pritisnite CTRL+D!\n");
118        printf("- Za nov racun unesite kod artikla:\n\n");
119        /* Unos bar koda provog artikla sledeceg kupca */
120        if (scanf("%ld", &kod) == EOF)
121            break;
122        /* Trenutni racun novog kupca */
123        racun = 0;
124        /* Za sve artikle trenutnog kupca */
125        while (1) {
126            /* Vrši se njihov pronalazak u nizu */
127            if ((i = binarna_pretraga(asortiman, n, kod)) == -1) {
128                printf
129                ("\tGreska: Ne postoji proizvod sa trazanim kodom!\n");
130            } else {
131                printf("\tTrazili ste:\t%s %s %12.2f\n",
132                    asortiman[i].naziv, asortiman[i].proizvodjac,
133                    asortiman[i].cena);
134                /* I dodaje se cena na ukupan racun */
135                racun += asortiman[i].cena;
136            }
137            /* Unos bar koda sledeceg artikla trenutnog kupca, ili 0 ako
138            on nema vise artikla */
139            printf("Unesite kod artikla [ili 0 za prekid]: \t");

```

```

140     scanf("%ld", &kod);
141     if (kod == 0)
142         break;
143 }
144 /* Stampa se ukupan racun trenutnog kupca */
145 printf("\n\tUKUPNO: %.2lf dinara.\n\n", racun);
146 }
147
148 printf("Kraj rada kase!\n");
149
150 exit(EXIT_SUCCESS);
151 }

```

Rešenje 1.24

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #define MAX 500
6
7  /* Struktura sa svim informacijama o pojedinacnom studentu */
8  typedef struct {
9      char ime[21];
10     char prezime[26];
11     int prisustvo;
12     int zadaci;
13 } Student;
14
15 /* Funkcija za sortiranje niza struktura po prezimenu
16     leksikografski rastuce */
17 void sort_ime_leksikografski(Student niz[], int n)
18 {
19     int i, j;
20     int min;
21     Student pom;
22
23     for (i = 0; i < n - 1; i++) {
24         min = i;
25         for (j = i + 1; j < n; j++)
26             if (strcmp(niz[j].ime, niz[min].ime) < 0)
27                 min = j;
28
29         if (min != i) {
30             pom = niz[min];
31             niz[min] = niz[i];
32             niz[i] = pom;
33         }
34     }
35 }

```

```

37 /* Funkcija za sortiranje niza struktura po ukupnom broju uradjenih
38    zadataka opadajuće, a ukoliko neki studenti imaju isti broj
39    uradjenih zadataka sortiraju se po dužini imena rastuće. */
40 void sort_zadatke_pa_imena(Student niz[], int n)
41 {
42     int i, j;
43     int max;
44     Student pom;
45     for (i = 0; i < n - 1; i++) {
46         max = i;
47         for (j = i + 1; j < n; j++)
48             if (niz[j].zadaci > niz[max].zadaci)
49                 max = j;
50         else if (niz[j].zadaci == niz[max].zadaci
51                 && strlen(niz[j].ime) < strlen(niz[max].ime))
52             max = j;
53         if (max != i) {
54             pom = niz[max];
55             niz[max] = niz[i];
56             niz[i] = pom;
57         }
58     }
59 }

60
61 /* Funkcija za sortiranje niza struktura opadajuće po broju casova
62    na kojima su bili. Ukoliko neki studenti imaju isti broj casova,
63    sortiraju se opadajuće po broju uradjenih zadataka, a ukoliko se
64    i po broju zadataka poklapaju, njihovo sortiranje ce biti po
65    prezimenu opadajuće. */
66 void sort_prisustvo_pa_zadatke_pa_prezimana(Student niz[], int n)
67 {
68     int i, j;
69     int max;
70     Student pom;
71     for (i = 0; i < n - 1; i++) {
72         max = i;
73         for (j = i + 1; j < n; j++)
74             if (niz[j].prisustvo > niz[max].prisustvo)
75                 max = j;
76             else if (niz[j].prisustvo == niz[max].prisustvo
77                     && niz[j].zadaci > niz[max].zadaci)
78                 max = j;
79             else if (niz[j].prisustvo == niz[max].prisustvo
80                     && niz[j].zadaci == niz[max].zadaci
81                     && strcmp(niz[j].prezime, niz[max].prezime) > 0)
82                 max = j;
83         if (max != i) {
84             pom = niz[max];
85             niz[max] = niz[i];
86             niz[i] = pom;
87         }
88     }
89 }

```

```

89 }
91 int main(int argc, char *argv[])
92 {
93     Student praktikum[MAX];
94     int i, br_studenata = 0;
95
96     FILE *fp = NULL;
97
98     /* Otvara se datoteka za citanje */
99     if ((fp = fopen("aktivnost.txt", "r")) == NULL) {
100         fprintf(stderr,
101             "Greska: Neupesno otvaranje datoteke aktivnost.txt.\n");
102         exit(EXIT_FAILURE);
103     }
104
105     /* Ucitava se sadrzaj */
106     for (i = 0;
107         fscanf(fp, "%s%d", praktikum[i].ime,
108             praktikum[i].prezime, &praktikum[i].prisustvo,
109             &praktikum[i].zadaci) != EOF; i++);
110     /* Zatvara se datoteka */
111     fclose(fp);
112     br_studenata = i;
113
114     /* Kreira se prvi spisak studenata po prvom kriterijumu */
115     sort_ime_leksikografski(praktikum, br_studenata);
116     /* Otvara se datoteka za pisanje */
117     if ((fp = fopen("dat1.txt", "w")) == NULL) {
118         fprintf(stderr,
119             "Greska: Neupesno otvaranje datoteke dat1.txt.\n");
120         exit(EXIT_FAILURE);
121     }
122     /* Upisuje se niz u datoteku */
123     fprintf
124     (fp, "Studenti sortirani po imenu leksikografski rastuce:\n");
125     for (i = 0; i < br_studenata; i++)
126         fprintf(fp, "%s %s %d %d\n", praktikum[i].ime,
127             praktikum[i].prezime, praktikum[i].prisustvo,
128             praktikum[i].zadaci);
129     /* Zatvara se datoteka */
130     fclose(fp);
131
132     /* Kreira se drugi spisak studenata po drugom kriterijumu */
133     sort_zadatke_pa_imena(praktikum, br_studenata);
134     /* Otvara se datoteka za pisanje */
135     if ((fp = fopen("dat2.txt", "w")) == NULL) {
136         fprintf(stderr,
137             "Greska: Neupesno otvaranje datoteke dat2.txt.\n");
138         exit(EXIT_FAILURE);
139     }
140     /* Upisuje se niz u datoteku */

```

```

141     fprintf(fp, "Studenti sortirani po broju zadataka opadajuće,\n");
142     fprintf(fp, "pa po dužini imena rastuće:\n");
143     for (i = 0; i < br_studenata; i++)
144         fprintf(fp, "%s %s %d %d\n", praktikum[i].ime,
145                 praktikum[i].prezime, praktikum[i].prisustvo,
146                 praktikum[i].zadaci);
147     /* Zatvara se datoteka */
148     fclose(fp);
149
150     /* Kreira se treci spisak studenata po trecem kriterijumu */
151     sort_prisustvo_pa_zadatke_pa_prezimenama(praktikum, br_studenata);
152     /* Otvara se datoteka za pisanje */
153     if ((fp = fopen("dat3.txt", "w")) == NULL) {
154         fprintf(stderr,
155                 "Greska: Neupesno otvaranje datoteke dat3.txt.\n");
156         exit(EXIT_FAILURE);
157     }
158     /* Upisuje se niz u datoteku */
159     fprintf(fp, "Studenti sortirani po prisustvu opadajuće,\n");
160     fprintf(fp, "pa po broju zadataka,\n");
161     fprintf(fp, "pa po prezimenima leksikografski opadajuće:\n");
162     for (i = 0; i < br_studenata; i++)
163         fprintf(fp, "%s %s %d %d\n", praktikum[i].ime,
164                 praktikum[i].prezime, praktikum[i].prisustvo,
165                 praktikum[i].zadaci);
166     /* Zatvara se datoteka */
167     fclose(fp);
168
169     exit(EXIT_SUCCESS);
170 }

```

Rešenje 1.25

```

#include <stdio.h>
2 #include <stdlib.h>
#include <string.h>
4
#define KORAK 10
6
/* Struktura koja opisuje jednu pesmu */
8 typedef struct {
    char *izvodjac;
10    char *naslov;
    int broj_gledanja;
12 } Pesma;

14 /* Funkcija za uporedjivanje pesama po broju gledanosti (potrebna
    za rad qsort funkcije) */
16 int uporedi_gledanost(const void *pp1, const void *pp2)
{
18     Pesma *p1 = (Pesma *) pp1;

```

```

    Pesma *p2 = (Pesma *) pp2;
20
    return p2->broj_gledanja - p1->broj_gledanja;
22 }

24 /* Funkcija za uporedjivanje pesama po naslovu (potrebna za rad
    qsort funkcije) */
26 int uporedi_naslove(const void *pp1, const void *pp2)
{
    Pesma *p1 = (Pesma *) pp1;
    Pesma *p2 = (Pesma *) pp2;

    return strcmp(p1->naslov, p2->naslov);
32 }

34 /* Funkcija za uporedjivanje pesama po izvodjacu (potrebna za rad
    qsort funkcije) */
36 int uporedi_izvodjace(const void *pp1, const void *pp2)
{
    Pesma *p1 = (Pesma *) pp1;
    Pesma *p2 = (Pesma *) pp2;

    return strcmp(p1->izvodjac, p2->izvodjac);
42 }

44 /* Funkcija koja oslobadja memoriju zauzetu dinamickim nizom pesme
    dimenzije n */
46 void oslobodi(Pesma * pesme, int n)
{
    int i;
    for (i = 0; i < n; i++) {
50         free(pesme[i].izvodjac);
        free(pesme[i].naslov);
52     }
    free(pesme);
54 }

56 int main(int argc, char *argv[])
{
    FILE *ulaz;
    /* Pokazivac na deo memorije za cuvanje pesama */
    Pesma *pesme;
    /* Broj mesta alociranih za pesme */
    int alocirano_zapesme;
    /* Redni broj pesme cije se informacije citaju */
    int i;
    /* Ukupan broj pesama */
    int n;
    int j;
    char c;
    /* Broj mesta alociranih za propratne informacije o pesmama */
70     int alocirano;

```

```

72     int broj_gledanja;

74     /* Priprema se datoteka za citanje */
75     ulaz = fopen("pesme.txt", "r");
76     if (ulaz == NULL) {
77         fprintf(stderr,
78             "Greska: Neuspesno otvaranje ulazne datoteke!\n");
79         exit(EXIT_FAILURE);
80     }

82     /* Citaju se informacije o pesmama */
83     pesme = NULL;
84     alocirano_za_pesme = 0;
85     i = 0;

86     while (1) {
87         /* Proverava se da li je dostignut kraj datoteke */
88         c = fgetc(ulaz);
89         if (c == EOF) {
90             /* Nema vise sadrzaja za citanje */
91             break;
92         } else {
93             /* Inace, vraca se procitani karakter nazad */
94             ungetc(c, ulaz);
95         }

96         /* Proverava se da li postoji dovoljno vec alocirane memorije
97            za citanje nove pesme */
98         if (alocirano_za_pesme == i) {
99             /* Ako ne, ako je potrosena sva alocirana memorija, alocira
100                se novih KORAK mesta */
101             alocirano_za_pesme += KORAK;
102             pesme =
103                 (Pesma *) realloc(pesme,
104                                     alocirano_za_pesme * sizeof(Pesma));

106             /* Proverava se da li je nova memorija uspesno realocirana */
107             if (pesme == NULL) {
108                 /* Ako nije ispisuje se obavestenje */
109                 fprintf(stderr,
110                     "Greska: Problem sa alokacijom memorije!\n");
111                 /* I oslobadja sva memorija zauzeta do ovog koraka */
112                 oslobodi(pesme, i);
113                 exit(EXIT_FAILURE);
114             }
115         }

116     }

118     /* Ako jeste, nastavlja se sa citanjem pesama ... */
119     /* Cita se ime izvodjaca */
120     /* Pozicija na koju treba smestiti procitani karakter */
121     j = 0;
122     /* Broj alociranih mesta */

```



```

124     alocirano = 0;
126     /* Memorija za smestanje procitanih karaktera */
    pesme[i].izvodjac = NULL;

128     /* Sve do prve beline u liniji (beline koja se nalazi nakon
    imena izvodjaca) citaju se karakteri iz datoteke */
    while ((c = fgetc(ulaz)) != ' ') {
130         /* Provera da li postoji dovoljno memorije za smestanje
        procitanog karaktera */
132         if (j == alocirano) {

134             /* Ako ne, ako je potrosena sva alocirana memorija, alocira
            se novih KORAK mesta */
136             alocirano += KORAK;
            pesme[i].izvodjac =
138                 (char *) realloc(pesme[i].izvodjac,
                                alocirano * sizeof(char));

140             /* Provera da li je nova alokacija uspesna */
142             if (pesme[i].izvodjac == NULL) {
                /* Ako nije oslobadja se sva memorija zauzeta do ovog
                koraka */
144                 oslobodi(pesme, i);
                /* I prekida sa izvršavanjem programa */
146                 exit(EXIT_FAILURE);
148             }
            }
150         /* Ako postoji dovoljno alocirane memorije, smesta se vec
        procitani karakter */
152         pesme[i].izvodjac[j] = c;
        j++;
154         /* I nastavlja se sa citanjem */
    }

156     /* Upis terminirajuće nule na kraj reci */
158     pesme[i].izvodjac[j] = '\0';

160     /* Preskace se karakter - */
    fgetc(ulaz);

162     /* Preskace se razmak */
164     fgetc(ulaz);

166     /* Cita se naslov pesme */
    /* Pozicija na koju treba smestiti procitani karakter */
168     j = 0;
    /* Broj alociranih mesta */
170     alocirano = 0;
    /* Memorija za smestanje procitanih karaktera */
172     pesme[i].naslov = NULL;

174     /* Sve do zareza (koji se nalazi nakon naslova pesme) citaju se

```

```

176     karakteri iz datoteke */
177 while ((c = fgetc(ulaz)) != ',') {
178     /* Provera da li postoji dovoljno memorije za smestanje
179        procitanog karaktera */
180     if (j == alocirano) {
181         /* Ako ne, ako je potrosena sva alocirana memorija, alocira
182            se novih KORAK mesta */
183         alocirano += KORAK;
184         pesme[i].naslov =
185             (char *) realloc(pesme[i].naslov,
186                             alocirano * sizeof(char));
187
188         /* Provera da li je nova alokacija uspesna */
189         if (pesme[i].naslov == NULL) {
190             /* Ako nije, oslobadja se sva memorija zauzeta do ovog
191                koraka */
192             free(pesme[i].izvodjac);
193             oslobodi(pesme, i);
194             /* I prekida izvorsavanje programa */
195             exit(EXIT_FAILURE);
196         }
197     }
198     /* Smesta se procitani karakter */
199     pesme[i].naslov[j] = c;
200     j++;
201     /* I nastavlja dalje sa citanjem */
202 }
203 /* Upisuje se terminirajuca nula na kraj reci */
204 pesme[i].naslov[j] = '\0';
205
206 /* Preskace se razmak */
207 fgetc(ulaz);
208
209 /* Cita se broj gledanja */
210 broj_gledanja = 0;
211
212 /* Sve do znaka za novi red (kraja linije) citaju se karakteri
213    iz datoteke */
214 while ((c = fgetc(ulaz)) != '\n') {
215     broj_gledanja = broj_gledanja * 10 + (c - '0');
216 }
217 pesme[i].broj_gledanja = broj_gledanja;
218
219 /* Prelazi se na citanje sledece pesme */
220 i++;
221 }
222
223 /* Informacija o broju procitanih pesama */
224 n = i;
225 /* Zatvara se datoteka */
226 fclose(ulaz);

```

```

228  /* Analiza argumenta komandne linije */
229  if (argc == 1) {
230      /* Nema dodatnih opcija => sortiranje po broju gledanja */
231      qsort(pesme, n, sizeof(Pesma), &uporedi_gledanost);
232  } else {
233      if (argc == 2 && strcmp(argv[1], "-n") == 0) {
234          /* Sortira se po naslovu */
235          qsort(pesme, n, sizeof(Pesma), &uporedi_naslove);
236      } else {
237          if (argc == 2 && strcmp(argv[1], "-i") == 0) {
238              /* Sortira se po izvodjacu */
239              qsort(pesme, n, sizeof(Pesma), &uporedi_izvodjace);
240          } else {
241              fprintf(stderr, "Greska: Nedoizvoljeni argumenti!\n");
242              free(pesme);
243              exit(EXIT_FAILURE);
244          }
245      }
246  }

247  /* Ispis rezultata */
248  for (i = 0; i < n; i++) {
249      printf("%s - %s, %d\n", pesme[i].izvodjac, pesme[i].naslov,
250             pesme[i].broj_gledanja);
251  }

252  /* Oslobadja se memorija */
253  oslobodi(pesme, n);

254  exit(EXIT_SUCCESS);
255  }

```

Rešenje 1.28

NAPOMENA: Rešenje koristi biblioteku za rad sa celobrojnim matricama iz zadatka ??.

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "matrica.h"
4
5  /* Funkcija odredjuje zbir v-te vrste matrice a sa m kolona */
6  int zbir_vrste(int **a, int v, int m)
7  {
8      int i, zbir = 0;
9
10     for (i = 0; i < m; i++) {
11         zbir += a[v][i];
12     }
13     return zbir;
14 }

```

```

16  /* Funkcija koja sortira vrste matrice (pokazivace na vrste) na
17     osnovu zbira elemenata koriscenjem selection sort algoritma */
18  void sortiraj_vrste(int **a, int n, int m)
19  {
20      int i, j, min;
21
22      for (i = 0; i < n - 1; i++) {
23          min = i;
24          for (j = i + 1; j < n; j++) {
25              if (zbir_vrste(a, j, m) < zbir_vrste(a, min, m)) {
26                  min = j;
27              }
28          }
29          if (min != i) {
30              int *tmp;
31              tmp = a[i];
32              a[i] = a[min];
33              a[min] = tmp;
34          }
35      }
36  }
37
38  int main(int argc, char *argv[])
39  {
40      int **a, n, m;
41
42      /* Unos dimenzija matrice */
43      printf("Unesite dimenzije matrice: ");
44      scanf("%d %d", &n, &m);
45
46      /* Alokacija memorije */
47      a = alociraj_matricu(n, m);
48      if (a == NULL) {
49          fprintf(stderr, "Greska: Neuspesna alokacija matrice\n");
50          exit(EXIT_FAILURE);
51      }
52
53      /* Ucitavaju se elementi matrice */
54      printf("Unesite elemente matrice po vrstama:\n");
55      ucitaj_matricu(a, n, m);
56
57      /* Poziva se funkcija koja sortira vrste matrice prema zbiru */
58      sortiraj_vrste(a, n, m);
59
60      /* Ispisuje se rezultujuca matrica */
61      printf("Sortirana matrica je:\n");
62      ispisi_matricu(a, n, m);
63
64      /* Oslobadja se memorija */
65      a = dealociraj_matricu(a, n);
66

```

```
        exit(EXIT_SUCCESS);
68 }

```

Rešenje 1.31

```
1  #include <stdio.h>
   #include <stdlib.h>
3  #include <math.h>
   #include <search.h>
5
   #define MAX 100
7
   /* Funkcija poredjenja dva cela broja (neopadajuci poredak) */
9  int poredi_int(const void *a, const void *b)
   {
11     /* Potrebno je konvertovati void pokazivace u int pokazivace koji
       se zatim dereferenciraju. */
13     int b1 = *((int *) a);
       int b2 = *((int *) b);
15
       /* Vrsi se poredjenje ovih vrednosti. */
17     if (b1 > b2)
         return 1;
19     else if (b1 < b2)
         return -1;
21     else
         return 0;
23
       /******
25         Umesto poredjenja, moze se koristiti naredba
           return b1 - b2;
27         Ipak, zbog moguceg prekoracenja prilikom oduzimanja, ovakvo
           resenje se koristi samo onda kada imamo ogranicene vrednosti
           promenljivih koje poredimo tj. kada ocekujemo da do
           prekoracenja ne moze da dodje.
           *****/
31     }
33
   /* Funkcija poredjenja dva cela broja (nerastuci poredak) */
35  int poredi_int_nerastuce(const void *a, const void *b)
   {
37     /* Za obrnuti poredak treba samo promeniti znak vrednosti koju
       koju vraca prethodna funkcija */
39     return -poredi_int(a, b);
   }
41
   int main()
43  {
       size_t n;
45     int i, x;
       int a[MAX], *p = NULL;

```

```

47  /* Unos dimenzije */
49  printf("Uneti dimenziju niza: ");
   scanf("%ld", &n);
51  if (n > MAX)
   n = MAX;
53
   /* Unos elementa niza */
55  printf("Uneti elemente niza:\n");
   for (i = 0; i < n; i++)
57     scanf("%d", &a[i]);
59
   /* Sortira se niz celih brojeva */
   qsort(a, n, sizeof(int), &poredi_int);
61
   /* Prikazuje se sortirani niz */
63  printf("Sortirani niz u rastucem poretku:\n");
   for (i = 0; i < n; i++)
65     printf("%d ", a[i]);
   putchar('\n');
67
   /* Pretrazuje se niz */
69  /* Vrednost koja ce biti trazena u nizu */
   printf("Uneti element koji se trazi u nizu: ");
71  scanf("%d", &x);
73
   /* Binarna pretraga */
   printf("Binarna pretraga: \n");
75  p = bsearch(&x, a, n, sizeof(int), &poredi_int);
   if (p == NULL)
77     printf("Elementa nema u nizu!\n");
   else
79     printf("Element je nadjen na poziciji %ld\n", p - a);
81
   /* Linearna pretraga */
   printf("Linearna pretraga (lfind): \n");
83  p = lfind(&x, a, &n, sizeof(int), &poredi_int);
   if (p == NULL)
85     printf("Elementa nema u nizu!\n");
   else
87     printf("Element je nadjen na poziciji %ld\n", p - a);
89  return 0;
   }

```

Rešenje 1.32

```

1  #include <stdio.h>
   #include <stdlib.h>
3  #include <math.h>
   #include <search.h>

```

```

5  #define MAX 100
7
9  /* Funkcija racuna broj delilaca broja x */
10 int broj_delilaca(int x)
11 {
12     int i;
13     int br;
14
15     /* Negativni brojevi imaju isti broj delilaca kao i pozitivni */
16     if (x < 0)
17         x = -x;
18     if (x == 0)
19         return 0;
20     if (x == 1)
21         return 1;
22     /* Svaki broj veci od 1 ima bar 2 delioca, (1 i samog sebe) */
23     br = 2;
24     for (i = 2; i < sqrt(x); i++)
25         if (x % i == 0)
26             /* Ako i deli x onda su delioci: i, x/i */
27             br += 2;
28     /* Ako je broj x bas kvadrat, onda se iz petlje izaslo kada je
29        promenljiva i bila bas jednaka korenu od x, i tada broj x ima
30        jos jednog delioca */
31     if (i * i == x)
32         br++;
33     return br;
34 }
35
36 /* Funkcija poredjenja dva cela broja po broju delilaca */
37 int poredi_po_broju_delilaca(const void *a, const void *b)
38 {
39     int ak = *(int *) a;
40     int bk = *(int *) b;
41     int n_d_a = broj_delilaca(ak);
42     int n_d_b = broj_delilaca(bk);
43
44     return n_d_a - n_d_b;
45 }
46
47 int main()
48 {
49     size_t n;
50     int i;
51     int a[MAX];
52
53     /* Unos dimenzije */
54     printf("Uneti dimenziju niza: ");
55     scanf("%ld", &n);
56     if (n > MAX)

```

```

57     n = MAX;

59     /* Unos elementa niza */
    printf("Uneti elemente niza:\n");
61     for (i = 0; i < n; i++)
        scanf("%d", &a[i]);

63

65     /* Sortira se niz celih brojeva prema broju delilaca */
    qsort(a, n, sizeof(int), &poredi_po_broju_delilaca);

67     /* Prikazuje se sortirani niz */
    printf
69     ("Sortirani niz u rastucem poretku prema broju delilaca:\n");
    for (i = 0; i < n; i++)
71         printf("%d ", a[i]);
    putchar('\n');

73

75     return 0;
}

```

Rešenje 1.33

```

1  #include <stdio.h>
   #include <stdlib.h>
3  #include <string.h>
   #include <search.h>
5
   #define MAX_NISKI 1000
7  #define MAX_DUZINA 31

9  /*****
   Niz nizova karaktera ovog potpisa
11  char niske[3][4];
   se moze graficki predstaviti ovako:
13  -----
   | a | b | c | \0 || d | e | \0|   || f | g | h | \0||
15  -----

   Dakle kao tri reci (abc, de, fgh), nadovezane jedna na drugu. Za
17  svaku je rezervisano po 4 karaktera ukljucujuci \0. Druga rec sa
   nalazi na adresi koja je za 4 veka od prve reci, a za 4 manja od
19  adrese na kojoj se nalazi treca rec. Adresa i-te reci je niske[i]
   i ona je tipa char*.

21

   Kako pokazivaci a i b u sledecoj funkciji sadrze adrese elemenata
23  koji trebaju biti uporedjeni, (npr. pri porecenju prve i poslednje
   reci, pokazivac a ce pokazivati na slovo 'a', a pokazivac b na
25  slovo 'f') treba ih kastovati na char*, i pozvati funkciju strcmp
   nad njima.

27  *****/
   int poredi_leksikografski(const void *a, const void *b)
29  {

```



```

31     return strcmp((char *) a, (char *) b);
32 }
33 /* Funkcija slicna prethodnoj, osim sto elemente ne uporedjuje
34    leksikografski, vec po duzini */
35 int poredi_duzine(const void *a, const void *b)
36 {
37     return strlen((char *) a) - strlen((char *) b);
38 }
39
40 int main()
41 {
42     int i;
43     size_t n;
44     FILE *fp = NULL;
45     char niske[MAX_NISKI][MAX_DUZINA];
46     char *p = NULL;
47     char x[MAX_DUZINA];
48
49     /* Otvara se datoteka */
50     if ((fp = fopen("niske.txt", "r")) == NULL) {
51         fprintf(stderr,
52             "Greska: Neupesno otvaranje datoteke niske.txt.\n");
53         exit(EXIT_FAILURE);
54     }
55
56     /* Cita se sadrzaj datoteke */
57     for (i = 0; fscanf(fp, "%s", niske[i]) != EOF; i++);
58
59     /* Zatvara se datoteka */
60     fclose(fp);
61     n = i;
62
63     /* Sortiraju se niske leksikografski. Biblioteckoj funkciji qsort
64        prosledjuje se funkcija kojom se zadaje kriterijum poredjenja 2
65        niske po duzini */
66     qsort(niske, n, MAX_DUZINA * sizeof(char),
67         &poredi_leksikografski);
68
69     printf("Leksikografski sortirane niske:\n");
70     for (i = 0; i < n; i++)
71         printf("%s ", niske[i]);
72     printf("\n");
73
74     /* Unosi se trazena niska */
75     printf("Uneti trazenu nisku: ");
76     scanf("%s", x);
77
78     /* Binarna pretraga */
79     /* Prosledjuje se pokazivac na funkciju poredi_leksikografski jer
80        je niz vec sortiran leksikografski. */
81     p = bsearch(&x, niske, n, MAX_DUZINA * sizeof(char),

```

```

83         &poredi_leksikografski);
84
85     if (p != NULL)
86         printf("Niska \"%s\" je pronadjena u nizu na poziciji %ld\n",
87                p, (p - (char *) niske) / MAX_DUZINA);
88     else
89         printf("Niska nije pronadjena u nizu\n");
90
91     /* Sortira se po duzini */
92     qsort(niske, n, MAX_DUZINA * sizeof(char), &poredi_duzine);
93
94     printf("Niske sortirane po duzini:\n");
95     for (i = 0; i < n; i++)
96         printf("%s ", niske[i]);
97     printf("\n");
98
99     /* Linearna pretraga */
100    p = lfind(&x, niske, &n, MAX_DUZINA * sizeof(char),
101              &poredi_leksikografski);
102    if (p != NULL)
103        printf("Niska \"%s\" je pronadjena u nizu na poziciji %ld\n",
104               p, (p - (char *) niske) / MAX_DUZINA);
105    else
106        printf("Niska nije pronadjena u nizu\n");
107
108    exit(EXIT_SUCCESS);
109 }

```

Rešenje 1.34

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <search.h>
5
6 #define MAX_NISKI 1000
7 #define MAX_DUZINA 31
8
9 /******
10  Niz pokazivaca na karaktere ovog potpisa
11  char *niske[3];
12  posle alokacije u main-u se moze graficki predstaviti ovako:
13
14  | X | -----> | a | b | c | \0|
15  | Y | -----> | d | e | \0|
16  | Z | -----> | f | g | h | \0|
17
18  Sa leve strane je vertikalno prikazan niz pokazivaca, gde je i-ti
19  njegov element pokazivac koji pokazuje na alocirane karaktere i-te
20

```

```

22     reci. Njegov tip je char*.

24     Kako pokazivaci a i b u sledecoj funkciji sadrze adrese elemenata
25     koji trebaju biti uporedjeni (recimo adresu od X i adresu od Z), i
26     kako su X i Z tipa char*, onda a i b su tipa char**, pa se tako
27     moraju i kastovati. Da bi se leksikografski uporedili elementi X i
28     Z, moraju se uporediti stringovi na koje oni pokazuju, pa zato se
29     u sledecoj funkciji poziva strcmp() nad onim na sta pokazuju a i b,
30     kastovani na odgovarajuci tip.
31     *****/
32     int poredi_leksikografski(const void *a, const void *b)
33     {
34         return strcmp(*(char **) a, *(char **) b);
35     }
36
37     /* Funkcija slicna prethodnoj, osim sto elemente ne uporedjuje
38        leksikografski, vec po duzini */
39     int poredi_duzine(const void *a, const void *b)
40     {
41         return strlen(*(char **) a) - strlen(*(char **) b);
42     }
43
44     /* Ovo je funkcija poredjenja za bsearch. Pokazivac b pokazuje na
45        element u nizu sa kojim se poredi, pa njega treba kastovati na
46        char** i dereferencirati, (videti obrazlozenje za prvu funkciju
47        u ovom zadatku, a pokazivac a pokazuje na element koji se trazi.
48        U main funkciji je to x, koji je tipa char*, tako da pokazivac a
49        ovde samo treba kastovati i ne dereferencirati. */
50     int poredi_leksikografski_b(const void *a, const void *b)
51     {
52         return strcmp((char *) a, *(char **) b);
53     }
54
55     int main()
56     {
57         int i;
58         size_t n;
59         FILE *fp = NULL;
60         char *niske[MAX_NISKI];
61         char **p = NULL;
62         char x[MAX_DUZINA];
63
64         /* Otvara se datoteka */
65         if ((fp = fopen("niske.txt", "r")) == NULL) {
66             fprintf(stderr,
67                     "Greska: Neuspesno otvaranje datoteke niske.txt.\n");
68             exit(EXIT_FAILURE);
69         }
70
71         /* Cita se sadrzaj datoteke */
72         i = 0;
73         while (fscanf(fp, "%s", x) != EOF) {

```

```

74      /* Alocira se dovoljno memorije za i-tu nisku */
      if ((niske[i] = malloc((strlen(x) + 1) * sizeof(char))) == NULL)
      {
76          fprintf(stderr, "Greska: Neuspesna alokacija niske\n");
          exit(EXIT_FAILURE);
78      }
      /* Kopira se procitana niska na svoje mesto */
80      strcpy(niske[i], x);
      i++;
82  }

84  /* Zatvara se datoteka */
  fclose(fp);
86  n = i;

88  /* Sortiraju se niske leksikografski. Biblioteckoj funkciji qsort
      se prosledjuje funkcija kojom se zadaje kriterijum poredjenja 2
90      niske po duzini */
  qsort(niske, n, sizeof(char *), &poredi_leksikografski);

92  printf("Leksikografski sortirane niske:\n");
94  for (i = 0; i < n; i++)
      printf("%s ", niske[i]);
96  printf("\n");

98  /* Unosi se trazena niska */
  printf("Uneti trazenu nisku: ");
100  scanf("%s", x);

102  /* Binarna pretraga */
  p = bsearch(x, niske, n, sizeof(char *),
104              &poredi_leksikografski_b);
  if (p != NULL)
106      printf("Niska \"%s\" je pronadjena u nizu na poziciji %ld\n",
              *p, p - niske);
108  else
      printf("Niska nije pronadjena u nizu\n");
110

112  /* Linearna pretraga */
  p = lfind(x, niske, &n, sizeof(char *), &poredi_leksikografski_b);
  if (p != NULL)
114      printf("Niska \"%s\" je pronadjena u nizu na poziciji %ld\n",
              *p, p - niske);
116  else
      printf("Niska nije pronadjena u nizu\n");
118

120  /* Sortira se po duzini */
  qsort(niske, n, sizeof(char *), &poredi_duzine);
  printf("Niske sortirane po duzini:\n");
122  for (i = 0; i < n; i++)
      printf("%s ", niske[i]);
124  printf("\n");

```

```

126     /* Oslobadja se zauzeta memorija */
127     for (i = 0; i < n; i++)
128         free(niske[i]);
129
130     exit(EXIT_SUCCESS);
131 }

```

Rešenje 1.35

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <search.h>
5
6  #define MAX 500
7
8  /* Struktura sa svim informacijama o pojedinacnom studentu */
9  typedef struct {
10     char ime[21];
11     char prezime[21];
12     int bodovi;
13 } Student;
14
15 /* Funkcija poredjenja za sortiranje po broju bodova. Studenti sa
16    istim brojem bodova se dodatno sortiraju leksikografski po
17    prezimenu */
18 int poredi1(const void *a, const void *b)
19 {
20     Student *prvi = (Student *) a;
21     Student *drugi = (Student *) b;
22
23     if (prvi->bodovi > drugi->bodovi)
24         return -1;
25     else if (prvi->bodovi < drugi->bodovi)
26         return 1;
27     else
28         /* Ako su jednaki po broju bodova, treba ih uporediti po
29            prezimenu */
30         return strcmp(prvi->prezime, drugi->prezime);
31 }
32
33 /* Funkcija za poredjenje koja se koristi u pretrazi po broju
34    bodova. Prvi parametar je ono sto se trazi u nizu (broj bodova),
35    a drugi parametar je element niza ciji se bodovi porede. */
36 int poredi2(const void *a, const void *b)
37 {
38     int bodovi = *(int *) a;
39     Student *s = (Student *) b;
40     return s->bodovi - bodovi;
41 }

```

```

42  /* Funkcija za poredjenje koja se koristi u pretrazi po prezimenu.
44     Prvi parametar je ono sto se trazi u nizu (prezime), a drugi
        parametar je element niza cije se prezime poredi. */
46  int poredi3(const void *a, const void *b)
47  {
48      char *prezime = (char *) a;
49      Student *s = (Student *) b;
50      return strcmp(prezime, s->prezime);
51  }
52
53  int main(int argc, char *argv[])
54  {
55      Student kolokvijum[MAX];
56      int i;
57      size_t br_studenata = 0;
58      Student *nadjen = NULL;
59      FILE *fp = NULL;
60      int bodovi;
61      char prezime[21];
62
63      /* Ako je program pozvan sa nedovoljnim brojem argumenata daje se
64         informacija korisniku kako se program koristi i prekida se
        izvršavanje. */
65      if (argc < 2) {
66          fprintf(stderr, "Greska: Program se poziva sa %s datoteka\n",
67                  argv[0]);
68          exit(EXIT_FAILURE);
69      }
70
71      /* Otvara se datoteka */
72      if ((fp = fopen(argv[1], "r")) == NULL) {
73          fprintf(stderr, "Greska: Neuspesno otvaranje datoteke %s\n",
74                  argv[1]);
75          exit(EXIT_FAILURE);
76      }
77
78      /* Ucitava se sadrzaj */
79      for (i = 0;
80           fscanf(fp, "%s%s%d", kolokvijum[i].ime,
81                  kolokvijum[i].prezime,
82                  &kolokvijum[i].bodovi) != EOF; i++);
83
84      /* Zatvara se datoteka */
85      fclose(fp);
86      br_studenata = i;
87
88      /* Sortira se niz studenata po broju bodova, gde se unutar grupe
89         studenata sa istim brojem bodova sortiranje vrši po prezimenu */
90      qsort(kolokvijum, br_studenata, sizeof(Student), &poredi1);
91
92      printf("Studenti sortirani po broju poena opadajuće, ");

```

```

94     printf("pa po prezimenu rastuce:\n");
    for (i = 0; i < br_studenata; i++)
96         printf("%s %s %d\n", kolokvijum[i].ime,
                kolokvijum[i].prezime, kolokvijum[i].bodovi);

98
100     /* Pretražuju se studenati po broju bodova binarnom pretragom jer
        je niz sortiran po broju bodova. */
    printf("Unesite broj bodova: ");
102     scanf("%d", &bodovi);

104     nadjen =
        bsearch(&bodovi, kolokvijum, br_studenata, sizeof(Student),
106               &poredi2);

108     if (nadjen != NULL)
        printf
110            ("Pronadjen je student sa unetim brojem bodova: %s %s %d\n",
             nadjen->ime, nadjen->prezime, nadjen->bodovi);
112     else
        printf("Nema studenta sa unetim brojem bodova\n");
114
116     /* Pretraga po prezimenu se mora vrsiti linearno jer je niz
        sortiran po bodovima. */
    printf("Unesite prezime: ");
118     scanf("%s", prezime);

120     nadjen =
        lfind(prezime, kolokvijum, &br_studenata, sizeof(Student),
122               &poredi3);

124     if (nadjen != NULL)
        printf
126            ("Pronadjen je student sa unetim prezimenom: %s %s %d\n",
             nadjen->ime, nadjen->prezime, nadjen->bodovi);
128     else
        printf("Nema studenta sa unetim prezimenom\n");
130
    exit(EXIT_SUCCESS);
132 }

```

Rešenje 1.36

```

#include <stdio.h>
2  #include <string.h>
#include <stdlib.h>

4
#define MAX 128

6
/* Funkcija poredi dva karaktera */
8  int uporedi_char(const void *pa, const void *pb)
{

```

```

10     return *(char *) pa - *(char *) pb;
11 }
12
13 /* Funkcija vraca 1 ako su argumenti anagrami, a 0 inace */
14 int anagrami(char s[], char t[])
15 {
16     /* Ako dve niske imaju razlicitu duzinu onda one nisu anagrami */
17     if (strlen(s) != strlen(t))
18         return 0;
19
20     /* Sortiraju se karakteri u niskama */
21     qsort(s, strlen(s) / sizeof(char), sizeof(char), &uporedi_char);
22     qsort(t, strlen(t) / sizeof(char), sizeof(char), &uporedi_char);
23
24     /* Ako su niske nakon sortiranja iste onda one jesu anagrami, u
25        suprotnom, nisu */
26     return !strcmp(s, t);
27 }
28
29 int main()
30 {
31     char s[MAX], t[MAX];
32
33     /* Unose se niske */
34     printf("Unesite prvu nisku: ");
35     scanf("%s", s);
36     printf("Unesite drugu nisku: ");
37     scanf("%s", t);
38
39     /* Ispituje se da li su niske anagrami */
40     if (anagrami(s, t))
41         printf("jesu\n");
42     else
43         printf("nisu\n");
44
45     return 0;
46 }

```

Rešenje 1.37

```

1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4
5  #define MAX 10
6  #define MAX_DUZINA 32
7
8  /* Funkcija poredjenja */
9  int uporedi_niske(const void *pa, const void *pb)
10 {
11     return strcmp((char *) pa, (char *) pb);

```



```

13 }
14
15 int main()
16 {
17     int i, n;
18     char S[MAX][MAX_DUZINA];
19
20     /* Unosi se broj niski */
21     printf("Unesite broj niski:");
22     scanf("%d", &n);
23
24     /* Unosi se niz niski */
25     printf("Unesite niske:\n");
26     for (i = 0; i < n; i++)
27         scanf("%s", S[i]);
28
29     /* Sortira se niz niski */
30     qsort(S, n, MAX_DUZINA * sizeof(char), &uporedi_niske);
31
32     /******
33     Ovaj deo je iskomentarisan jer se u zadatku ne trazi ispis
34     sortiranih niski. Koriscen je samo u fazi testiranja programa.
35     *****/
36
37     printf("Sortirane niske su:\n");
38     for(i = 0; i < n; i++)
39         printf("%s ", S[i]);
40
41     /******
42     Ako postoje dve iste niske u nizu, onda ce one nakon
43     sortiranja niza biti jedna do druge */
44     for (i = 0; i < n - 1; i++)
45         if (strcmp(S[i], S[i + 1]) == 0) {
46             printf("ima\n");
47             return 0;
48         }
49     printf("nema\n");
50
51     return 0;
52 }

```

Rešenje 1.38

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #define MAX 21
6
7  /* Struktura koja predstavlja jednog studenta */
8  typedef struct student {
9      char nalog[8];

```

```

11     char ime[MAX];
12     char prezime[MAX];
13     int poeni;
14 } Student;

15 /* Funkcija poredi studente prema broju poena, rastuce */
16 int uporedi_poeni(const void *a, const void *b)
17 {
18     Student s = *(Student *) a;
19     Student t = *(Student *) b;
20     return s.poeni - t.poeni;
21 }

22 /* Funkcija poredi studente prvo prema godini, zatim prema smeru i
23    na kraju prema indeksu */
24 int uporedi_nalog(const void *a, const void *b)
25 {
26     Student s = *(Student *) a;
27     Student t = *(Student *) b;
28     /* Za svakog studenta iz naloga se izdvaja godina upisa, smer i
29        broj indeksa */
30     int godina1 = (s.nalog[2] - '0') * 10 + s.nalog[3] - '0';
31     int godina2 = (t.nalog[2] - '0') * 10 + t.nalog[3] - '0';
32     char smer1 = s.nalog[1];
33     char smer2 = t.nalog[1];
34     int indeks1 =
35         (s.nalog[4] - '0') * 100 + (s.nalog[5] - '0') * 10 +
36         s.nalog[6] - '0';
37     int indeks2 =
38         (t.nalog[4] - '0') * 100 + (t.nalog[5] - '0') * 10 +
39         t.nalog[6] - '0';
40     if (godina1 != godina2)
41         return godina1 - godina2;
42     else if (smer1 != smer2)
43         return smer1 - smer2;
44     else
45         return indeks1 - indeks2;
46 }

47

48 /* Funkcija poredjenja po nalogu za upotrebu u bibliotečkoj
49    funkciji bsearch */
50 int uporedi_bsearch(const void *a, const void *b)
51 {
52     /* Nalog studenta koji se traži */
53     char *nalog = (char *) a;
54     /* Ključ pretrage */
55     Student s = *(Student *) b;
56
57     int godina1 = (nalog[2] - '0') * 10 + nalog[3] - '0';
58     int godina2 = (s.nalog[2] - '0') * 10 + s.nalog[3] - '0';
59     char smer1 = nalog[1];
60     char smer2 = s.nalog[1];

```

```

63     int indeks1 =
        (nalog[4] - '0') * 100 + (nalog[5] - '0') * 10 + (nalog[6] -
64                                     '0');
65     int indeks2 =
        (s.nalog[4] - '0') * 100 + (s.nalog[5] - '0') * 10 +
66         (s.nalog[6] - '0');
67     if (godina1 != godina2)
68         return godina1 - godina2;
69     else if (smer1 != smer2)
70         return smer1 - smer2;
71     else
72         return indeks1 - indeks2;
73 }
74
75 int main(int argc, char **argv)
76 {
77     Student *nadjen = NULL;
78     char nalog_trazeni[8];
79     Student niz_studenata[100];
80     int i = 0, br_studenata = 0;
81     FILE *in = NULL, *out = NULL;
82
83     /* Ako je broj argumenata komandne linije razlicit i od 2 i od 3,
84        korisnik nije ispravno pokrenuo program. */
85     if (argc != 2 && argc != 3) {
86         fprintf(stderr,
87             "Greska: Program se poziva sa %s -opcija [nalog]\n",
88             argv[0]);
89         exit(EXIT_FAILURE);
90     }
91
92     /* Otvara se datoteka za citanje */
93     in = fopen("studenti.txt", "r");
94     if (in == NULL) {
95         fprintf(stderr,
96             "Greska: Neuspesno otvaranje datoteke studenti.txt!\n");
97         exit(EXIT_FAILURE);
98     }
99
100     /* Otvara se datoteka za pisanje */
101     out = fopen("izlaz.txt", "w");
102     if (out == NULL) {
103         fprintf(stderr,
104             "Greska: Neuspesno otvaranje datoteke izlaz.txt!\n");
105         exit(EXIT_FAILURE);
106     }
107
108     /* Ucitavaju se studenti iz ulazne datoteke sve do njenog kraja */
109     while (fscanf
110         (in, "%s %s %s %d", niz_studenata[i].nalog,
111          niz_studenata[i].ime, niz_studenata[i].prezime,
112          &niz_studenata[i].poeni) != EOF)

```

```

115         i++;
117
118     br_studenata = i;
119
120     /* Ako je prisutna opcija -p, vrsi se sortiranje po poenima */
121     if (strcmp(argv[1], "-p") == 0)
122         qsort(niz_studenata, br_studenata, sizeof(Student),
123             &uporedi_poeni);
124     /* A ako je prisutna opcija -n, vrsi se sortiranje po nalogu */
125     else if (strcmp(argv[1], "-n") == 0)
126         qsort(niz_studenata, br_studenata, sizeof(Student),
127             &uporedi_nalog);
128
129     /* Sortirani studenti se ispisuju u izlaznu datoteku */
130     for (i = 0; i < br_studenata; i++)
131         fprintf(out, "%s %s %s %d\n", niz_studenata[i].nalog,
132             niz_studenata[i].ime, niz_studenata[i].prezime,
133             niz_studenata[i].poeni);
134
135     /* Ukoliko je u komandnoj liniji uz opciju -n naveden i nalog
136        studenta... */
137     if (argc == 3 && (strcmp(argv[1], "-n") == 0)) {
138         strcpy(nalog_trazeni, argv[2]);
139
140         /* ... pronalazi se student sa tim nalogom. */
141         nadjen =
142             (Student *) bsearch(nalog_trazeni, niz_studenata,
143                 br_studenata, sizeof(Student),
144                 &uporedi_bsearch);
145
146         if (nadjen == NULL)
147             printf("Nije nadjen!\n");
148         else
149             printf("%s %s %s %d\n", nadjen->nalog, nadjen->ime,
150                 nadjen->prezime, nadjen->poeni);
151     }
152
153     /* Zatvaraju se datoteke */
154     fclose(in);
155     fclose(out);
156
157     exit(EXIT_SUCCESS);
158 }

```