Univerzitet u Beogradu
Matematički fakultet

Milena, Jelena, Ana, Mirko, Anđelka, Nina

Zbirka programa

Beograd, 2015.

Predgovor

U okviru kursa *Programiranje 2* na Matematičkom fakultetu vežbaju se zadaci koji imaju za cilj da studente nauče rekurzivnom pristupu rešavanju problema, ispravan rad sa pokazivačima i dinamički alociranom memorijom, osnovne algoritme pretraživanja i sortiranja, kao i rad 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 kolokvijuma i ispita.

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, pomenimo tu, pre svega, Milana Bankovića i doc dr Filipa Marića. Zbog toga smatramo da je naš osnovni doprinos što smo objedinili, precizno formulisali i rešili sve najvažnije zadatke koji su potrebni za uspešno savlađivanje koncepata koji se obrađuju u okviru kursa.

Zahvaljujemo se recenzentima na ..., kao i studentima koji su svojim aktivnim učešćem u nastavi pomogli i doprineli u obličavanju ovog materijala.

Autori

Sadržaj

1 Uvodni zadaci **3**
- 1.1 Podela koda po datotekama . . . 3
- 1.2 Bitovi . . . 3
- 1.3 Rekurzija . . . 9
- 1.4 Rešenja . . . 12

2 Pokazivači **15**
- 2.1 Pokazivači i aritmetika sa pokazivačima . . . 15
- 2.2 Višedimenzioni nizovi . . . 19
- 2.3 Dinamička alokacija memorije . . . 23
- 2.4 Pokazivači na funkcije . . . 26
- 2.5 Rešenja . . . 27

3 Algoritmi pretrage i sortiranja **35**
- 3.1 Pretraživanje . . . 35
- 3.2 Sortiranje . . . 38
- 3.3 Bibliotečke funkcije pretrage i sortiranja . . . 44
- 3.4 Rešenja . . . 46

4 Dinamičke strukture podataka **57**
- 4.1 Liste . . . 57
- 4.2 Drveta . . . 63
- 4.3 Rešenja . . . 63

5 Ispitni rokovi **67**
- 5.1 Programiranje 2, praktični deo ispita, jun 2015. . . . 67
- 5.2 Programiranje 2, praktični deo ispita, jul 2015. . . . 68
- 5.3 Rešenja . . . 70

Literatura **76**

Glava 1

Uvodni zadaci

1.1 Podela koda po datotekama

1.2 Algoritmi za rad sa bitovima

Zadatak 1.1

(a) Napisati funkciju `print_bits` koja štampa bitove u binarnom zapisu celog broja x.

(b) Napisati program koja testira funkciju `print_bits` za brojeve koji se sa standardnog ulaza zadaju u heksadekasnom formatu.

Test 1

```
Ulaz:   0x7F
Izlaz:  0000 0000 0000 0000 0000 0000 0111 1111
```

Test 2

```
Ulaz:   0x80
Izlaz:  0000 0000 0000 0000 0000 0000 1000 0000
```

Test 3

```
Ulaz:   0x00FF00FF
Izlaz:  0000 0000 1111 1111 0000 0000 1111 1111
```

Test 4

```
Ulaz:   0xFFFFFFFF
Izlaz:  1111 1111 1111 1111 1111 1111 1111 1111
```

Test

```
Ulaz:   0xABCDE123
Izlaz:  1010 1011 1100 1101 1110 0001 0010 0011
```

Zadatak 1.2 Napisati funkciju koja broji bitove postavljene na `1` u zapisu broja x. Napisati program koji testira tu funkciju za brojeve koji se sa standardnog ulaza zadaju u heksadekasnom formatu.

Test 1

```
Ulaz:   0x7F
Izlaz:  7
```

Test 2

```
Ulaz:   0x80
Izlaz:  1
```

Test 3

```
Ulaz:   0x00FF00FF
Izlaz:  16
```

Test 4

```
Ulaz:   0xFFFFFFFF
Izlaz:  32
```

Test 4

```
Ulaz:   0xABCDE123
Izlaz:  17
```

Zadatak 1.3

(a) Napisati funkciju `najveci` koja određuje najveći broj koji se može zapisati istim binarnim ciframa kao dati broj i funkciju `najmanji` koja određuje najmanji broj koji se može zapisati istim binarnim ciframa kao dati broj.

(b) Napisati program koji testira prethodno napisane funkcije tako što prikazuje binarnu reprezentaciju brojeva koji se dobijaju nakon poziva funkcije `najveci`, ondosno `najmanji` za brojeve koji se sa standardnog ulaza zadaju u heksadekasnom formatu.

Test 1

```
Ulaz:   0x7F
Izlaz:
 Najveci:
 1111 1110 0000 0000 0000 0000 0000 0000
 Najmanji:
 0000 0000 0000 0000 0000 0000 0111 1110
```

Test 2

```
Ulaz:   0x80
Izlaz:
 Najveci:
 1000 0000 0000 0000 0000 0000 0000 0000
 Najmanji:
 0000 0000 0000 0000 0000 0000 0000 0001
```

Test 3

```
Ulaz:   0x00FF00FF
Izlaz:
 Najveci:
 1111 1111 1111 1111 0000 0000 0000 0000
 Najmanji:
 0000 0000 0000 0000 1111 1111 1111 1111
```

Test 4

```
Ulaz:   0xFFFFFFFF
Izlaz:
 Najveci:
 1111 1111 1111 1111 1111 1111 1111 1111
 Najmanji:
 1111 1111 1111 1111 1111 1111 1111 1111
```

Test 4

```
Ulaz:   0xABCDE123
Izlaz:
 Najveci:
 1111 1111 1111 1111 1000 0000 0000 0000
 Najmanji:
 0000 0000 0000 0001 1111 1111 1111 1111
```

Zadatak 1.4 Napisati program za rad sa bitovima.

(a) Napisati funkciju funkciju koja određuje broj koji se dobija kada se n bitova datog broja, počevši od pozicije p postave na `0`.

(b) Napisati funkciju koja određuje broj koji se dobija kada se n bitova datog broja, počevši od pozicije p postave na `1`.

(c) Napisati funkciju koja određuje broj koji se dobija kada se n bitova datog broja, počevši od pozicije p i vraća ih kao bitove najmanje težine rezultata.

(d) Napisati funkciju koja vraća broj koji se dobija upisivanjem poslednjih n bitova broja y u broj x, počevši od pozicije p.

(e) Napisati funkciju koja vraća broj koji se dobija invertovanjem n bitova broja x počevši od pozicije p.

(f) Napisati program koji testira prethodno napisane funkcije.

Program treba da testira prethodno napisane funkcije nad neoznačenim celim brojem koji se unosi sa standardnog ulaza. *Napomena: Pozicije se broje počev od pozicije najnižeg bita, pri čemu se broji od nule .*

Zadatak 1.5

(a) Napisati funkciju funkciju koja određuje broj koji se dobija rotiranjem u levo datog celog broja. Napomena: Rotiranje podrazumeva pomeranje svih bitova za jednu poziciju ulevo, s tim što se bit sa pozicije najviše težine pomera na mesto najmanje težine.

(b) Napisati funkciju koja određuje broj koji se dobija rotiranjem u desno datog celog neoznačenog broja.

(c) Napisati funkciju koja određuje broj koji se dobija rotiranjem u desno datog celog broja.

(d) Napisati program koji testira prethodno napisane funkcije za brojeve koji se sa standardnog ulaza zadaju u heksadekasnom formatu.

Zadatak 1.6 Napisati funkciju `mirror` koja određuje ceo broj čiji binarni zapis je slika u ogledalu binarnog zapisa argumenta funkcije. Napisati i program koji testira datu funkciju za brojeve koji se sa standardnog ulaza zadaju u heksadekasnom formatu, tako što najpre ispisuje binarnu reprezentaciju unetog broja, a potom i binarnu reprezentaciju broja dobijenog nakon poziva funkcije `mirror` za uneti broj.

Zadatak 1.7 Napisati funkciju `int Broj01(unsigned int n)` koja za dati broj `n` vraća `1` ako u njegovom binarnom zapisu ima više jednica nego nula, a inače vraća `0`. Napisati program koji tu funkciju testira za broj koji se zadaje sa standardnog ulaza.

Test 1

```
Ulaz:   10
Izlaz:  0
```

Test 2

```
Ulaz:   1024
Izlaz:  0
```

Test 3

```
Ulaz:   2147377146
Izlaz:  1
```

Test 4

```
Ulaz:   1111111115
Izlaz:  0
```

Zadatak 1.8 Napisati funkciju koja broji koliko se puta kombinacija `11` (dve uzastopne jedinice) pojavljuje u binarnom zapisu celog neoznačenog broja x. Tri uzastopne jedinice se broje dva puta. Napisati program koji tu funkciju testira za broj koji se zadaje sa standardnog ulaza.

Test 1

```
Ulaz:   11
Izlaz:  1
```

Test 2

```
Ulaz:   1024
Izlaz:  0
```

Test 3

```
Ulaz:   2147377146
Izlaz:  22
```

Test 4

```
Ulaz:   1111111115
Izlaz:  6
```

Zadatak 1.9 Napisati program koji sa standardnog ulaza učitava pozitivan ceo broj, a na standardni izlaz ispisuje vrednost tog broja sa razmenjenim vrednostima bitova na pozicijama i, j. Pozicije i, j se učitavaju kao parametri komandne linije. Smatrati da je krajnji desni bit binarne reprezentacije `0`-ti bit. Pri rešavanju nije dozvoljeno koristiti pomoćni niz niti aritmetičke operatore +,-,/,*,%.

Test 1

```
Poziv:  ./a.out 1 2
Ulaz:   11
Izlaz:  13
```

Test 2

```
Poziv: ./a.out 1 2
Ulaz:   1024
Izlaz:  1024
```

Test 3

```
Poziv: ./a.out 12 12
Ulaz:   12345
Izlaz:  12345
```

Zadatak 1.10 Napisati funkciju koja na osnovu neoznačenog broja x formira nisku s koja sadrži heksadekadni zapis broja x, koristeći algoritam za brzo prevođenje binarnog u heksadekadni zapis (svake 4 binarne cifre se zamenjuju jednom odgovarajućom heksadekadnom cifrom). Napisati program koji tu funkciju testira za broj koji se zadaje sa standardnog ulaza.

Test 1

```
Ulaz:   11
Izlaz:  0000000B
```

Test 2

```
Ulaz:  1024
Izlaz: 00000400
```

Test 3

```
Ulaz:  12345
Izlaz: 00003039
```

Zadatak 1.11 Napisati funkciju koja za dva data neoznačena broja x i y invertuje u podatku x one bitove koji se poklapaju sa odgovarajućim bitovima u broju y. Ostali bitovi ostaju nepromenjeni. Napisati program koji tu funkciju testira za brojeve koji se zadaju sa standardnog ulaza.

Test 1

```
Ulaz:   123 10
Izlaz:  4294967285
```

Test 2

```
Ulaz:  3251 0
Izlaz: 4294967295
```

Test 3

```
Ulaz:   12541 1024
Izlaz:  4294966271
```

Zadatak 1.12 Napisati funkciju koja računa koliko petica bi imao ceo neoznačen broj x u oktalnom zapisu. Napisati program koji tu funkciju testira za broj koji se zadaje sa standardnog ulaza.

Test 1

```
Ulaz:   123
Izlaz:  0
```

Test 2

```
Ulaz:   3245
Izlaz:  2
```

Test 3

```
Ulaz:   100328
Izlaz:  1
```

Milena: Naredne zadatke prebaciti da budu nakon rekurzije.

Zadatak 1.13 Napisati rekurzivnu funkciju vraća broj bitova koji su postavljeni na 1, u binarnoj reprezentaciji njenog celobrojnog argumenta. Napisati program koji testira napisanu funkciju za brojeve koji se učitavaju sa standardnog ulaza zadati u heksadekadnom formatu.

Test 1

```
Ulaz:   0x7F
Izlaz:  7
```

Test 2

```
Ulaz:   0x80
Izlaz:  1
```

Test 3

```
Ulaz:   0x00FF00FF
Izlaz:  16
```

Test 4

```
Ulaz:   0xFFFFFFFF
Izlaz:  32
```

Zadatak 1.14

Napisati rekurzivnu funkciju koja štampa bitovsku reprezentaciju neoznačenog celog broja, i program koji je testira za ulaz koji se zadaje sa standardnog ulaza.

Test 1

```
Ulaz:        10
Izlaz:       00000000000000000000000000001010
```

Zadatak 1.15 Nina: Ovo je prebaceno iz poglavlja sa pokazivacima. Napisati rekurzivnu funkciju za određivanje najveće cifre u oktalnom zapisu neoznačenog celog broja korišćenjem bitskih operatora. Uputstvo: binarne cifre grupisati u podgrupe od po tri cifre, počev od bitova najmanje težine.

Test 1

```
Ulaz:   5
Izlaz:  5
```

Test 2

```
Ulaz:   125
Izlaz:  7
```

Test 3

```
Ulaz:   8
Izlaz:  1
```

Test 4

```
Ulaz:   10
Izlaz:  2
```

Zadatak 1.16 Nina: Ovo je prebaceno iz poglavlja sa pokazivacima. Napisati rekurzivnu funkciju za određivanje (dekadne vrednosti) najveće cifre u heksadekadnom zapisu neoznačenog celog broja korišćenjem bitskih operatora. Uputstvo: binarne cifre grupisati u podgrupe od po četiri cifre, počev od bitova najmanje težine.

Test 1

```
Ulaz:   5
Izlaz:  5
```

Test 2

```
Ulaz:   16
Izlaz:  1
```

Test 3

```
Ulaz:   18
Izlaz:  2
```

Test 4

```
Ulaz:   165
Izlaz:  10
```

1.3 Rekurzija

Zadatak 1.17 Napisati rekurzivnu funkciju koja sumira elemente niza celih brojeva. Napisati program koji testira napisane funkcije. Sa standardnog ulaza učitati dimenziju n ($0 < n \leq 100$) celobrojong niza, a zatim i elemente niza. Na standardni izlaz ispisati rezultat primene napisane funkcije nad učitanim nizom.

Zadatak 1.18 Napisati rekurzivnu funkciju koja izračunava x^k, za dati realni broj x i prirodan broj k. Napisati program koji testira napisanu funkciju za vrednosti koje se unose sa standardnog ulaza.

Zadatak 1.19 Koristeći uzajamnu (posrednu) rekurziju napisati naredne dve funkcije:

- fukciju `paran` koja proverava da li je broj cifara nekog broja paran i vraća 1 ako jeste, a 0 inače;
- i fukciju `neparan` koja vraća 1 ukoliko je broj cifara nekog broja neparan, a 0 inače.

Napisati program koji testira napisanu funkciju tako što se za heksadekadnu vrednost koja se unosi sa standardnog ulaza ispisuje da li je paran ili neparan.

Zadatak 1.20 Napisati repno-rekurzivnu funkciju koja izračunava faktorijel broja n. Napisati program koji testira napisanu funkciju za poizvoljan broj n ($n \leq 12$) unet sa standardnog ulaza.

Zadatak 1.21 Elementi funkcije F izračunavaju se na osnovu sledećih rekurentnih relacija:

$$F(0) = 0$$

$$F(1) = 1$$

$$F(n) = a * F(n-1) + b * F(n-2)$$

(a) Napisati rekurzivnu funkciju koja računa n-ti element u nizu F.

(b) Napisati rekurzivnu funkciju koja računa n-ti element u nizu F ali tako da se problemi manje dimenzije rešavaju samo jedan put.

Napisati program koji testira napisane funkcije za poizvoljan broj n ($n \in \mathbb{N}$) unet sa standardnog ulaza.

Zadatak 1.22 Napisati rekurzivnu funkciju koja prikazuje sve permutacije skupa $\{1, 2, ..., n\}$. Napisati program koji testira napisanu funkciju za poizvoljan prirodan broj n ($n \leq 50$) unet sa standardnog ulaza.

Zadatak 1.23 Paskalov trougao se dobija tako što mu je svako polje (izuzev jedinica po krajevima) zbir jednog polja levo i jednog polja iznad.

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5   10  10  5   1
```

(a) Napisati rekurzivnu funkciju koja izračunava d_n kao sumu elemenata n-te hipotenuze Paskalovog trougla.

(b) Napisati rekurzivnu funkciju koja izračunava vrednost polja `(i, j)`. Milena: dodati sta je i a sta j tj odakle se broji

Zadatak 1.24 Napisati rekurzivnu funkciju koja određuje maksimum niza celih brojeva. Napisati program koji testira ovu funkciju, za niz koji se unosi sa standardnog ulaza. Niz neće imati više od 256 elemenata, i njegovi elementi se unose sve do kraja ulaza.

Test 1

```
Ulaz:   3 2 1 4 21
Izlaz:  21
```

Test 2

```
Ulaz:   2 -1 0 -5 -10
Izlaz:  2
```

Test 3

```
Ulaz:   1 11 3 5 8 1
Izlaz:  11
```

Test 4

```
Ulaz:   5
Izlaz:  5
```

Zadatak 1.25 Napisati rekurzivnu funkciju koja izračunava skalarni proizvod dva data vektora. Napisati program koji testira ovu funkciju, za nizove koji se unose sa standardnog ulaza. Nizovi neće imati više od 256 elemenata. Prvo se unosi dimenzija nizova, a zatim i sami njihovi elementi.

Test 1

```
Ulaz:   3 1 2 3 1 2 3
Izlaz:  14
```

Test 2

```
Ulaz:   2 3 5 2 6
Izlaz:  36
```

Test 3

```
Ulaz:   0
Izlaz:  0
```

Zadatak 1.26 Napisati rekurzivnu funkciju koja sabira dekadne cifre datog celog broja x. Napisati program koji testira ovu funkciju, za broj koji se unosi sa standardnog ulaza.

Test 1

```
Ulaz:   123
Izlaz:  6
```

Test 2

```
Ulaz:   23156
Izlaz:  17
```

Test 3

```
Ulaz:   1432
Izlaz:  10
```

Test 4

```
Ulaz:   1
Izlaz:  1
```

Test 5

```
Ulaz:   0
Izlaz:  0
```

Zadatak 1.27 Napisati rekurzivnu funkciju koja računa broj pojavljivanja elementa x u nizu a dužine n. Napisati program koji testira ovu funkciju, za x i niz koji se unose sa standardnog ulaza. Niz neće imati više od 256 elemenata. Prvo se unosi x, a zatim elementi niza sve do kraja ulaza.

Test 1

```
Ulaz:   4 1 2 3 4
Izlaz:  1
```

Test 2

```
Ulaz:   11 3 2 11 14 11 43 1
Izlaz:  2
```

Test 3

```
Ulaz:  1 3 21 5 6
Izlaz:  0
```

Zadatak 1.28 Napisati rekurzivnu funkciju koja ispituje da li je data niska palindrom. Napisati program koji testira ovu funkciju. Pretposatviti da niska neće neće imati više od 31 karaktera, i da se unosi sa standardnog ulaza.

Test 1

```
Ulaz:   programiranje
Izlaz:  ne
```

Test 2

```
Ulaz:   anavolimilovana
Izlaz:  da
```

Test 3

```
Ulaz:   a
Izlaz:  da
```

Test 4

```
Ulaz:   aba
Izlaz:  da
```

Test

```
Ulaz:   aa
Izlaz:  da
```

Zadatak 1.29 Napisati rekurzivnu funkciju kojom se proverava da li su tri zadata broja uzastopni članovi niza. Potom, napisati program koji je testira. Sa standardnog ulaza se unose najpre tri tražena broja, a zatim elementi niza, sve do kraja ulaza. Pretpostaviti da neće biti uneto više od 256 brojeva.

Test 1

```
Ulaz:   1 2 3 4 1 2 3 4 5
Izlaz:  da
```

Test 2

```
Ulaz:   1 2 3 11 1 2 4 3 6
Izlaz:  ne
```

Test 3

```
Ulaz:   1 2 3 1 2
Izlaz:  ne
```

Zadatak 1.30 Napisati rekurzivnu funkciju koja prikazuje sve varijacije sa

ponavljanjem dužine n skupa $\{a, b\}$, i program koji je testira, za n koje se unosi sa standardnog ulaza.

Test 1

```
Ulaz:    3
Izlaz:   a a a
         a a b
         a b a
         a b b
         b a a
         b a b
         b b a
         b b b
```

Zadatak 1.31 *Hanojske kule*: Data su tri vertikalna štapa, na jednom se nalazi n diskova poluprečnika 1,2,3,... do n, tako da se najveći nalazi na dnu, a najmanji na vrhu. Ostala dva štapa su prazna. Potrebno je premestiti diskove na drugi štap tako da budu u istom redosledu, pri čemu se ni u jednom trenutku ne sme staviti veći disk preko manjeg, a preostali štap se koristi kao pomoćni štap prilikom premeštanja.
Napisati program koji za proizvoljnu vrednost n, koja se unosi sa standardnog ulaza, prikazuje proces premeštanja diskova.

Zadatak 1.32 *Modifikacija Hanojskih kula*: Data su četiri vertikalna štapa, na jednom se nalazi n diskova poluprečnika 1,2,3,... do n, tako da se najveći nalazi na dnu, a najmanji na vrhu. Ostala tri štapa su prazna. Potrebno je premestiti diskove na drugi štap tako da budu u istom redosledu, premestajući jedan po jedan disk, pri čemu se ni u jednom trenutku ne sme staviti veći disk preko manjeg, pri čemu se preostala dva štapa koriste kao pomoćni štapovi prilikom premeštanja.
Napisati program koji za proizvoljnu vrednost n, koja se unosi sa standardnog ulaza, prikazuje proces premeštanja diskova.

1.4 Rešenja

Rešenje 1.1

Rešenje 1.2

Rešenje 1.3

Rešenje 1.4

Rešenje 1.5

Rešenje 1.6

Rešenje 1.7

Rešenje 1.8

Rešenje 1.9

Rešenje 1.10

Rešenje 1.11

Rešenje 1.12

Rešenje 1.13

```
#include<stdio.h>

int main(){
  printf("Hello bitovi!\n"); /* Da li komentari rade čćžš*/
  return 0;
}
```

Rešenje 1.15

Rešenje 1.16

Rešenje 1.17

```
#include<stdio.h>

int main(){
  printf("Hello bitovi!\n"); /* Da li komentari rade čćžš*/
  return 0;
}
```

Rešenje 1.18

```
#include<stdio.h>

int main(){
  printf("Hello bitovi!\n"); /* Da li komentari rade čćžš*/
  return 0;
}
```

Rešenje 1.19

Rešenje 1.20

Rešenje 1.21

Rešenje 1.22

Rešenje 1.23

Rešenje 1.24

Rešenje 1.25

Rešenje 1.26

Rešenje 1.27

Rešenje 1.28

Rešenje 1.29

Glava 2

Pokazivači

2.1 Pokazivači i aritmetika sa pokazivačima

Zadatak 2.1 Za dati celobrojni niz dimenzije n, napisati funkcije koje obrću njegove elemente:

(a) korišćenjem indeksne sintakse,

(b) korišćenjem pokazivačke sintakse.

Napisati program koji testira napisane funkcije. Sa standardnog ulaza učitati dimenziju niza n ($n \leq 100$), a zatim elemente niza. Prikazati sadržaj niza pre i posle poziva funkcije za obrtanje elemenata niza.

Zadatak 2.2 Dat je celobrojni niz dimenzije n.

(a) Napisati funkciju `zbir` koja izračunava zbir elemenata niza.

(b) Napisati funkciju `proizvod` koja izračunava proizvod elemenata niza.

(c) Napisati funkciju `min_element` koja izračunava najmanji elemenat niza.

(d) Napisati funkciju `max_element` koja izračunava najveći elemenat niza.

Napisati program koji testira napisane funkcije. Sa standardnog ulaza učitati dimenziju n ($0 < n \leq 100$) celobrojong niza, a zatim i elemente niza. Na standardni izlaz ispisati rezultat primene svake od napisanih funkcija nad učitanim nizom.

Zadatak 2.3 Korišćenjem pokazivačke sintakse, napisati funkciju koja vrednosti elemenata u prvoj polovini niza povećava za jedan, a u drugoj polovini smanjuje za jedan. Ukoliko niz ima neparan broj elemenata, onda vrednost srednjeg elementa niza ostaviti nepromenjenim. Napisati program koji testira napisanu funkciju. Sa standardnog ulaza učitati dimenziju n ($0 \leq n \leq 100$) celobrojong niza, a zatim i elemente niza. Na standardni izlaz ispisati rezultat primene svake od napisanih funkcija nad učitanim nizom.

Zadatak 2.4 Napisati program koji ispisuje broj prihvaćenih argumenata komandne linije, a zatim i same argumenate kojima prethode njihovi redni brojevi.

Zadatak 2.5 Korišćenjem pokazivačke sintakse, napisati funkciju koja za datu nisku ispituje da li je palindrom. Napisati program koji vrši prebrojavanje argumenata komandne linije koji su palindromi.

Test 1

```
Poziv:  ./a.out programiranje anavolimilovana topot ana anagram t
Izlaz:  4
```

Zadatak 2.6 Napisati program koji kao prvi argument komandne linije prihvata putanju do datoteke za koju treba proveriti koliko reči ima n karaktera, gde se n zadaje kao drugi argument komandne linije. U zadatku ne koristiti ugrađene funkcije za rad sa niskama, već implementirati svoje koristeći pokazivačku sintaksu.

Test 1

```
Poziv:    ./a.out fajl.txt 1
Datoteka: Ovo je sadrzaj
   datoteke i u
          njoj ima reci koje
   imaju
          1 karakter
Izlaz:    2
```

Zadatak 2.7 Napisati program koji kao prvi argument komandne linije prihvata putanju do datoteke za koju treba proveriti koliko reči ima zadati sufiks (ili prefiks), koji se zadaje kao drugi argument komandne linije. Program je neophodno pozvati sa jednom od opcija `-s` ili `-p` u zavisnosti od čega treba proveriti koliko reči ima zadati sufiks (ili prefiks). U zadatku ne koristiti ugrađene funkcije za rad sa niskama, već implementirati svoje koristeći pokazivačku sintaksu.

Test 1

```
Poziv:    ./a.out fajl.txt ke -
   s
Datoteka: Ovo je sadrzaj
   datoteke
          i u njoj ima reci
   koje se
          zavrsavaju na ke
Izlaz:    2
```

Zadatak 2.8 Milena: Ovo bi trebalo da ide u pretraživanje/sortiranje? Napisati funkciju koja u rastuće sortiranom nizu celih brojeva binarnom pretragom pronalazi prvi element veći od nule i kao rezultat vraća njegovu poziciju u nizu. Ukoliko nema elemenata većih od nule, funkcija kao rezultat vraća `-1`. Napisati program koji testira ovu funkciju za niz elemenata koji se učitavaju sa standardnog ulaza. Niz neće biti duži od 256, i njegovi elementi se unose sve do kraja ulaza.

Test 1

```
Ulaz:   -151 -44 5
        12 13 15
Izlaz: 2
```

Test 2

```
Ulaz:   -100 -15 -11
        -8 -7 -5
Izlaz: -1
```

Test 3

```
Ulaz:   -100 -15 0 13
        155 124 258
        315 516 7000
Izlaz: 3
```

Zadatak 2.9 Milena: Ovo bi trebalo da ide u pretraživanje/sortiranje? Napisati funkciju koja u opadajuće sortiranom nizu celih brojeva binarnom pretragom pronalazi prvi element manji od nule i kao rezultat vraća njegovu poziciju u nizu. Ukoliko nema elemenata manjih od nule, funkcija kao rezultat vraća `-1`. Napisati program koji testira ovu funkciju za niz elemenata koji se učitavaju sa standardnog ulaza. Niz neće biti duži od 256, i njegovi elementi se unose sve do kraja ulaza.

Test 1

```
Ulaz:   151 44 5 -12 -13 -15
Izlaz: 3
```

Test 2

```
Ulaz:  100 55 15 0 -15 -124
   -155
       -258 -315 -516 -7000
Izlaz: 4
```

Test 3

```
Ulaz:   100 15 11 8 7 5 4 3 2
Izlaz: -1
```

Zadatak 2.10 Milena: Ovo bi trebalo da ide u pretraživanje/sortiranje? Struktura `Student` čuva podatke o broju indeksa studenta i poenima sa kolokvijuma, pri čemu su brojevi indeksa i poeni sa kolokvijuma celi brojevi. Napisati program koji učitava podatke o studentima iz datoteke „kolokvijum.txt“ u kojoj se nalazi najviše 500 zapisa o studentima. Sortirati ovaj niz studenata po broju poena opadajuće, a ako više studenata ima isti broj poena, onda po broju indeksa rastuće. Ispisati sortiran niz studenata na standardni izlaz.

Test 1

```
Kolokvijum.txt:    20140015 25
                   20140115 24
                   20130250 3
                   20140001 4
                   20140038 25
Izlaz:             20140015 25
                   20140038 25
                   20140115 24
                   20140001 4
                   20130250 3
```

Test 2

```
Kolokvijum.txt:    20140015 25
                   20110010 12
                   20140105 0
                   20120110 13
Izlaz:             20140015 25
                   20120110 13
                   20110010 12
                   20140105 0
```

Zadatak 2.11 Napisati strukturu `Student` koja čuva podatke o broju indeksa studenta i broju poena osvojenih na testu. Pretpostaviti da su brojevi indeksa celi brojevi, a poeni sa testa realni brojevi. Napisati program koji učitava podatke o studentima iz datoteke „studenti.txt“ u kojoj se nalazi najviše 100 zapisa o studentima. Sortirati ovaj niz studenata po broju poena rastuće, a ako više studenata ima isti broj poena, onda po broju indeksa opadajuće. Ispisati sortiran niz studenata na standardni izlaz.

Test 1

```
Kolokvijum.txt:    20140015 4.5
                   20130115 4.5
                   20140250 3.4
                   20110304 1.2
Izlaz:             20110304 1.2
                   20140250 3.4
                   20140015 4.5
                   20130115 4.5
```

Test 2

```
Kolokvijum.txt:    20130015 9.5
                   20130010 9.5
                   20090103 0.5
                   20140005 10.0
                   20140120 1.3
                   20140038 2.5
Izlaz:             20090103 0.5
                   20140120 1.3
                   20140038 2.5
                   20130015 9.5
                   20130010 9.5
                   20140005 10.0
```

2.2 Višedimenzioni nizovi

Zadatak 2.12 Data je kvadratna matrica dimenzije n.

(a) Napisati funkciju koja izračunava trag matrice.

(b) Napisati funkciju koja izračunava euklidsku normu matrice.

(c) Napisati funkciju koja izračunava gornju vandijagonalnu normu matrice.

Napisati program koji testira napisane funkcije. Sa standardnog ulaza učitati dimanziju kvadratne matrice n ($0 < n \leq 100$), a zatim i elemente matrice. Na standardni izlaz ispisati rezultat primene svake od napisanih funkcija nad učitanom matricom.

Zadatak 2.13 Date su dve kvadratne matrice istih dimenzija n.

(a) Napisati funkciju koja proverava da li su matrice jednake.

(b) Napisati funkciju koja izračunava zbir matrica.

(c) Napisati funkciju koja izračunava proizvod matrica.

Napisati program koji testira napisane funkcije. Sa standardnog ulaza učitati dimanziju kvadratnih matrica n ($0 < n \leq 100$), a zatim i elemente matrica. Na standardni izlaz ispisati rezultat primene svake od napisanih funkcija nad učitanim matricama.

Test 1

```
Ulaz:   3
        1 2 3
        1 2 3
        1 2 3
        1 2 3
        1 2 3
        1 2 3
Izlaz:  da
        2 4 6
        2 4 6
        2 4 6
        6 12 18
        6 12 18
        6 12 18
```

Zadatak 2.14 Relacija se može predstaviti kvadratnom matricom nula i jedinica na sledeći način: dva elementa i i j su u relaciji ukoliko se u preseku i-te vrste i j-te kolone matrice nalazi broj 1, a nisu u relaciji ukoliko se tu nalazi broj 0.

(a) Napisati funkciju koja proverava da li je relacija zadata matricom refleksivna.

(b) Napisati funkciju koja proverava da li je relacija zadata matricom simetrična.

(c) Napisati funkciju koja proverava da li je relacija zadata matricom tranzitivna.

(d) Napisati funkciju koja određuje refleksivno zatvorenje relacije (najmanju refleksivnu relaciju koja sadrži datu).

(e) Napisati funkciju koja određuje simetrično zatvorenje relacije (najmanju simetričnu relaciju koja sadrži datu).

(f) Napisati funkciju koja određuje refleksivno-tranzitivno zatvorenje relacije (najmanju refleksivnu i tranzitivnu relaciju koja sadrži datu)(Napomena: koristiti Varšalov algoritam).

Napisati program koji učitava matricu iz datoteke čije se ime zadaje kao prvi argument komandne linije. U prvoj liniji datoteke nalazi se dimenzija matrice n ($0 < n \leq 64$), a potom i sami elementi matrice. Na standardni izlaz ispisati rezultat testiranja napisanih funkcija.

Test 1

```
Datoteka:  4
           1 0 0 0
           0 1 1 0
           0 0 1 0
           0 0 0 0
Izlaz:     refleksivnost: ne
           simetricnost: ne
           tranzitivnost: da
           refleksivno zatvorenje:
           1 0 0 0
           0 1 1 0
           0 0 1 0
           0 0 0 1
           simetricno zatvorenje:
           1 0 0 0
           0 1 1 0
           0 1 1 0
           0 0 0 0
           refleksivno-tranzitivno zatvorenje:
           1 0 0 0
           0 1 1 0
           0 0 1 0
           0 0 0 1
```

Zadatak 2.15 Data je kvadratna matrica dimenzije n.

(a) Napisati funkciju koja određuje najveći element matrice na sporednoj dijagonali.

(b) Napisati funkciju koja određuje indeks kolone koja sadrži najmanji element matrice.

(c) Napisati funkciju koja određuje indeks vrste koja sadrži najveći element matrice.

(d) Napisati funkciju koja određuje broj negativnih elemenata matrice.

Napisati program koji testira napisane funkcije. Sa standardnog ulaza učitati elemente celobrojne kvadratne matrice čija se dimenzija n ($0 < n \leq 32$) zadaje kao argument komandne linije.

Test 1

```
Poziv:  ./a.out 3
Ulaz:   1 2 3
        -4 -5 -6
        7 8 9
Izlaz:  7 2 2 3
```

Zadatak 2.16 Napisati funkciju kojom se proverava da li je zadata kvadratna matrica dimenzije n ortonormirana. Matrica je ortonormirana ako je skalarni proizvod svakog para različitih vrsta jednak nuli, a skalarni proizvod vrste sa samom sobom jednak jedinici. Napisati program koji testira napisanu funkciju. Sa standardnog ulaza učitati dimenziju celobrojne kvadratne matrice n ($n \leq 32$), a zatim i njene elemente. Na standardni izlaz ispisati rezultat primene napisane funkcije na učitanu matricu.

Test 1

```
Ulaz:  4
       1 0 0 0
       0 1 0 0
       0 0 1 0
       0 0 0 1
Izlaz: da
```

Test 2

```
Ulaz:  3
       1 2 3
       5 6 7
       1 4 2
Izlaz: ne
```

Zadatak 2.17 Data je matrica dimenzije $n \times m$.

(a) Napsiati funkciju koja učitava elemente matrice sa standardnog ulaza

(b) Napsiati funkciju koja na standardni izlaz spiralno ispisuje elemente matrice.

Napisati program koji testira napisane funkcije. Sa standardnog ulaza učitati dimenzije matrice n ($0 < n \leq 10$) i m ($0 < n \leq 10$), a zatim i elemente matrice (pozivom gore napisane funkcije). Na standardni izlaz spiralno ispisati elemente učitane matrice.

Zadatak 2.18 Milena: Ovo bi trebalo da ide u pretraživanje/sortiranje? Napisati funkciju koja vrši leksikografsko opadajuće sortiranje niza niski (pretpostaviti da ima najvise 1000 niski, od kojih svaka ima najvise 30 karaktera). Napisati program koji testira rad napisane funkcije. Niske učitati iz datoteke „niske.txt“ (prva linija datoteke sadrži broj niski, a svaka sledeća po jednu nisku). Na standardni izlaz ispisati leksikografski opadajuće sortirane niske.

Zadatak 2.19 Napisati funkciju koja izračunava k-ti stepen kvadratne matrice dimenzije n ($n \leq 32$). Napisati program koji testira napisanu funkciju. Sa standardnog ulaza učitati dimenziju celobrojne matrice n, elemente matrice i stepen k ($0 < k \leq 10$). Na standardni izlaz ispisati rezultat primene napisane funkcije. Napomena: voditi računa da se prilikom stepenovanja matrice izvrši što manji broj množenja.

Test 1

```
Ulaz:  3
       1 2 3
       4 5 6
       7 8 9
       8
Izlaz: 510008400 626654232
   743300064
       1154967822 1419124617
   1683281412
       1799927244 2211595002
   2623262760
```

2.3 Dinamička alokacija memorije

Zadatak 2.20 Napisati program koji sa standardnog ulaza učitava niz celih brojeva a zatim na standardni izlaz ispisuje ove brojeve u obrnutom poretku. Ne praviti nikakavo ograničenje za dimenziju niza.

Zadatak 2.21 Napisati funkciju koja kao rezultat vraća nisku koja se dobija nadovezivanjem dve niske, bez promene njihovog sadržaja. Napisati program koji testira rad napisane funkcije. Sa standardnog ulaza učitati dve niske karaktera. Na standardni izlaz ispisati nisku koja se dobija njihovim nadovezivanjem.

Zadatak 2.22 Napisati program koji sa standardnog ulaza učitava niz celih brojeva. Brojevi se unose sve dok se ne unese nula. Ne praviti nikakve pretpostavke o dimenziji niza. Na standardni izlaz ispisati ovaj niz brojeva u obrnutom poretku. Zadatak uraditi na dva načina:

(a) realokaciju memorije niza vršiti korišćenjem `malloc()` funkcije,

(b) realokaciju memorije niza vršiti korišćenjem `realloc()` funkcije.

Zadatak 2.23 Napisati program koji sa standardnog ulaza učitava matricu celih brojeva. Prvo se učitavaju dimenzije matrice n i m (ne praviti nikakve pretpostavke o njihovoj veličini), a zatim i elementi matrice. Na standardni izlaz ispisati trag matrice.

Zadatak 2.24 Data je celobrojna matrica dimenzije $n \times m$ napisati:

(a) Napisati funkciju koja vrši učitavanje matrice sa standardnog ulaza.

(b) Napisati funkciju koja ispisuje elemente ispod glavne dijagonale matrice.

Napisati program koji testira napisane funkcije. Sa standardnog ulaza učitati n i m (ne praviti nikakve pretpostavke o njihovoj veličini), zatim učitati elemente matrice i na standardni izlaz ispisati elemente ispod glavne dijagonale matrice.

Zadatak 2.25 Milena: Ovo bi trebalo da ide u pretraživanje/sortiranje? U datoteci „pesme.txt“ nalaze se informacije o gledanosti pesama na Youtube-u. Format datoteke sa informacijama je sledeći:

- U prvoj liniji datoteke može i ne mora da se nalazi ukupan broj pesama prisutnih u datoteci.
- Svaki naredni red datoteke sadrži informacije o gledanosti pesama u formatu `izvodjac - naslov, brojGledanja`.

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 sortirane na opisan način.

(a) Uraditi zadatak uz pretpostavku da je maksimalna dužina imena izvođača 20 karaktera, a imena naslova pesme 50 kraraktera.

(b) Uraditi zadatak bez pravljenja pretpostavki o maksimalnoj dužini imena izvođača i naslova pesme.

Test 1

```
Poziv: ./a.out
Datoteka:  5
           Ana - Nebo, 2342
           Laza - Oblaci, 29
           Pera - Ptice, 327
           Jelena - Sunce,
   92321
           Mika - Kisa, 5341
Izlaz:     Jelena - Sunce,
   92321
           Mika - Kisa, 5341
           Ana - Nebo, 2342
           Pera - Ptice, 327
           Laza - Oblaci, 29
```

Zadatak 2.26 Za zadatu matricu dimenzije $n \times m$ napisati funkciju koja izračunava redni broj kolone matrice čiji je zbir maksimalan. Napisati program koji testira ovu funkciju. Sa standardnog ulaza učitati dimenzije matrice n i m, a zatim elemente matrice.

Zadatak 2.27 Za zadatu kvadratnu matricu dimenzije n napisati funkciju koja menja njen sadržaj tako što polovi elemente iznad glavne dijagonale, duplira elemente ispod glavne dijagonale, dok elemente na glavnoj dijagonali ostavlja nepromenjene. Napisati program koji testira ovu funkciju za vrednosti koje se učitavaju iz datoteke „matrica.txt“. U datoteci se nalazi prvo dimenzija matrice, a zatim redom elementi matrice.

Zadatak 2.28 Za zadatu kvadratnu matricu dimenzije n napisati funkciju koja izračunava zbir apsolutnih vrednosti matrice ispod sporedne dijagonale. Napisati program koji testira ovu funkciju za vrednosti koje se učitavaju iz datoteke čije se ime zadaje kao argument komandne linije. U datoteci se nalazi prvo dimenzija matrice, a zatim redom elementi matrice.

Zadatak 2.29 Za zadatu kvadratnu matricu dimenzije n napisati funkciju koja vrši sortiranje vrsta matrice, rastuće na osnovu sume elemenata u vrsti. Napisati program koji testira ovu funkciju. Sa standardnog ulaza se prvo unosi dimenzija matrice, a zatim redom elementi matrice. Rezultujuću matricu ispisati na standardni izlaz.

Zadatak 2.30 Petar sakuplja sličice igrača za predstojeće Svetsko prvenstvo u fudbalu. U datoteci „slicice.txt“ se nalaze informacije o sličicama koje mu nedostaju u formatu: `redni_broj_sličice ime_reprezentacije_kojoj_sličica_pripada`. Pomozite Petru da otkrije koliko mu sličica ukupno nedostaje, kao i da pronađe ime reprezentacije čijih sličica ima najmanje. Dobijene podatke ispisati na standardni izlaz. Napomena: za realokaciju memorije koristiti `realloc()` funkciju.

Zadatak 2.31 U datoteci „temena.txt“ se nalaze tačke koje predstavljaju temena nekog n-tougla. Napisati program koji na osnovu sadržaja datoteke na standardni izlaz ispisuje o kom n-touglu je reč, a zatim i vrednosti njegovog obima i površine. Pretpostavka je da će mnogougao biti konveksan.

Zadatak 2.32 Napisati program koji na osnovu dve matrice dimenzija $m \times n$ formira matricu dimenzije $2 \cdot m \times n$ tako što naizmenično kombinuje jednu vrstu prve matrice i jednu vrstu druge matrice. Matirce su zapisane u datoteci „matrice.txt“. U prvom redu se nalaze dimenzije matrica m i n, u narednih m redova se nalaze vrste prve matrice, a u narednih m redova vrste druge matrice. Rezultujuću matricu ispisati na standardni izlaz.

Zadatak 2.33 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 standardni izlaz.

Zadatak 2.34 Na ulazu se zadaje niz celih brojeva čiji se unos završava nulom. Napisati funkciju koja od zadatog niza formira matricu tako da prva vrsta odgovara unetom nizu, a svaka naredna se dobija cikličkim pomeranjem elemenata niza za jednu poziciju ulevo.

Napisati program koji testira ovu funkciju. Sa standardnog ulaza se prvo unosi

dimenzija matrice, a zatim redom elementi matrice. Rezultujuću matricu ispisati na standardni izlaz.

2.4 Pokazivači na funkcije

Zadatak 2.35 Napisati program koji tabelarno štampa vrednosti proizvoljne realne funkcije sa jednim realnim argumentom, odnosno izračunava i ispisuje vrednosti date funkcije na diskretnoj ekvidistantnoj mreži od n tačaka intervala $[a, b]$. Realni brojevi a i b ($a < b$) kao i ceo broj n ($n \geq 2$) se učitavaju sa standardnog ulaza. Ime funkcije se zadaje kao argument komandne linije (`sin`, `cos`, `tan`, `atan`, `acos`, `asin`, `exp`, `log`, `log10`, `sqrt`, `floor`, `ceil`, `sqr`).

Zadatak 2.36 Napisati funkciju koja izračunava limes funkcije `f(x)` u tački a. Adresa funkcije `f` čiji se limes računa se prenosi kao parametar funkciji za računanje limesa. Limes se računa sledećom aproksimacijom (vrednosti n i a uneti sa standardnog ulaza kao i ime funkcije):

$$lim_{x \to a} f(x) = lim_{n \to \infty} f(a + \frac{1}{n})$$

Test 1

```
Ulaz:   tan 1.570795 10000
Izlaz:  -10134.5
```

Test 2

```
Ulaz:   log 0 1000000
Izlaz:  -13.81551
```

Zadatak 2.37 Napisati funkciju koja određuje integral funkcije `f(x)` na intervalu $[a, b]$. Adresa funkcije `f` se prenosi kao parametar. Integral se računa prema formuli:

$$\int_a^b f(x) = h \cdot (\frac{f(a) + f(b)}{2} + \Sigma_{i=1}^{n} f(a + i \cdot h))$$

Vrednost h se izračunava po formuli $h = (b - a)/n$, dok se vrednosti n, a i b unose sa standardnog ulaza kao i ime funkcije iz zaglavlja `math.h`. Na standardni izlaz ispisati vrednost integrala.

Zadatak 2.38 Napisati funkciju koja približno izračunava integral funkcije `f(x)` na intervalu $[a, b]$. Funkcija `f` se prosleđuje kao parametar, a integral se procenjuje po Simpsonovoj formuli:

$$I = \frac{h}{3} \left(f(a) + 4 \sum_{i=1}^{n/2} f(a + (2i - 1)h) + 2 \sum_{i=1}^{n/2-1} f(a + 2ih) + f(b) \right)$$

Granice intervala i n su argumenti funkcije. Napisati program, koji kao argumente komandne linije prihvata ime funkcije iz zaglavlja `math.h`, krajeve intervala pretrage i n, a na standardni izlaz ispisuje vrednost odgovarajućeg integrala.

2.5 Rešenja

Rešenje 2.1

```c
#include<stdio.h>

int main(){
  printf("Hello bitovi!\n");
  return 0;
}
```

Rešenje 2.2

```c
#include<stdio.h>

int main(){
  printf("Hello bitovi!\n");
  return 0;
}
```

Rešenje 2.3

```c
#include<stdio.h>

int main(){
  printf("Hello bitovi!\n");
  return 0;
}
```

Rešenje 2.4

```c
#include<stdio.h>

int main(){
  printf("Hello bitovi!\n");
  return 0;
}
```

Rešenje 2.5

```c
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.6

```c
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.7

```c
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.8

```c
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.9

```c
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.10

```c
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.11

```c
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.12

```
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.13

```
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.14

```
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.15

```
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.16

```
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.17

```
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
```

```
  return 0;
}
```

Rešenje 2.18

```
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.19

```
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.20

```
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.21

```
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.22

```
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.23

```c
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.24

```c
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.25

```c
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.26

```c
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.27

```c
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.28

```c
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.29

```c
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.30

```c
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.31

```c
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.32

```c
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.33

```c
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.34

```c
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
```

```
  return 0;
}
```

Rešenje 2.35

```
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.36

```
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.37

```
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Rešenje 2.38

```
#include<stdio.h>

int main(){
  printf("Hello pokazivaci!\n");
  return 0;
}
```

Glava 3

Algoritmi pretrage i sortiranja

3.1 Pretraživanje

Zadatak 3.1 Napisati iterativne funkcije pretraga nizova. Svaka funkcija treba da vrati indeks pozicije na kojoj je pronađen traženi element ili broj -1 ukoliko element nije pronađen.

(a) Napisati funkciju koja vrši linearnu pretragu niza celih brojeva `a`, dužine `n`, tražeći u njemu broj `x`.

(b) Napisati funkciju koja vrši binarnu pretragu sortiranog niza `a`, dužine `n`, tražeći u njemu broj `x`.

(c) Napisati funkciju koja vrši interpoacionu pretragu sortiranog niza `a`, dužine `n`, tražeći u njemu broj `x`.

Napisati i program koji generiše slučajni rastući niz dimenzije `n` (prvi argument komandne linije), i u njemu već napisanim funkcijama traži element `x` (drugi argument komandne linije). Potrebna vremena za izvršavanje ovih funkcija upisati u fajl `vremena.txt`.

Test 1

```
Poziv: ./a.out 1000000 235423
Izlaz:
Linearna pretraga
Element nije u nizu
---------------------------------
Binarna pretraga
Element nije u nizu
---------------------------------
Interpolaciona pretraga
Element nije u nizu
---------------------------------
```

Zadatak 3.2 Napisati rekurzivne funkcije algoritama linearne, binarne i interpolacione pretrage i program koji ih testira za brojeve koji se unose sa standardnog

ulaza. Pretpostaviti da niz brojeva koji se unosi neće biti duži od 1024 elemenata. Prvo se unosi broj koji se traži, a zatim sortirani elementi niza sve do kraja ulaza.

Test 1

```
Ulaz: 11 2 5 6 8 10 11 23
Izlaz:
Linearna pretraga
Pozicija elementa je 5.
Binarna pretraga
Pozicija elementa je 5.
Interpolaciona pretraga
Pozicija elementa je 5.
```

Test 2

```
Ulaz: 14 10 32 35 43 66 89 100
Izlaz:
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 3.3 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 studenta čije informacije se potom prikazuju na ekranu. Zatim, korisnik učitava prezime studenta i prikazuju mu se informacije o prvom studentu sa unetim prezimenom. Pretrage implementirati u vidu iterativnih funkcija što bolje manje složenosti. Pretpostaviti da u datoteci neće biti više od 128 studenata, i da su imena i prezimena svih kraća od 16 slova.

Test 1

```
Datoteka:
20140003 Marina Petrovic
20140012 Stefan Mitrovic
20140032 Dejan Popovic
20140049 Mirko Brankovic
20140076 Sonja Stevanovic
20140104 Ivan Popovic
20140187 Vlada Stankovic
20140234 Darko Brankovic

Ulaz:           Izlaz:

20140076        Indeks: 20140076, Ime i prezime: Sonja Stevanovic
Popovic         Indeks: 20140032, Ime i prezime: Dejan Popovic
```

Zadatak 3.4 Modifikovati prethodni zadatak 3.3 tako da tražene funkcije budu rekurzivne.

Zadatak 3.5 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 datateci veći od 0 i ne veći od 1024.

Test 1

```
Poziv: ./a.out dat.txt -x
Datoteka:
12 53
2.342 34.1
-0.3 23
-1 23.1
123.5 756.12
Izlaz: -0.3 23
```

Zadatak 3.6 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: korisiti algoritam analogan algoritmu binarne pretrage.

Test 1

```
Izlaz:
1.57031
```

Zadatak 3.7 Napisati funkciju koja u sortiranom nizu nalazi prvi element veći od 0. Napisati i program koji testira ovu funkciju za niz elemenata koji se zadaju kao argumenti komandne linije. Uputstvo: primeniti binarnu pretragu.

Test 1

```
Poziv:  ./a.out -43 -24 -5 -2 1
    4 6 12
Izlaz:  1
```

Test 2

```
Poziv:  ./a.out -32 4 65 123
Izlaz:  4
```

Zadatak 3.8 Napisati funkciju koja određuje ceo deo logaritma za osnovu 2 datog neoznačenog celog broja, koristeći samo bitske i relacione operatore.

(a) Napisati funkciju, linearne složenosti, koja određuje logaritam pomeranjem broja udesno dok ne postane 0.

(b) Napisati funkciju, logaritmske složenosti, koja određuje logaritam koristeći binarnu pretragu.

Tražene funkcije testirati programom koji broj učitava sa standardnog ulaza, a logaritam ispisuje na standardni izlaz.

Test 1

```
Ulaz:    10
Izlaz:   3 3
```

Test 2

```
Ulaz:    4
Izlaz:   2 2
```

Test 3

```
Ulaz:    17
Izlaz:   4 3
```

Test 4

```
Ulaz:    1031
Izlaz:   10 10
```

**** Zadatak 3.9** U prvom kvadrantu dato je $1 \leq \mathtt{N} \leq 10000$ duži svojim koordinatama (duži mogu da se seku, preklapaju, itd.). Napisati program koji pronalazi najmanji ugao $0 \leq \alpha \leq 90°$, 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 $[0, 90°]$.

Test 1

```
Ulaz:
2
2 0 2 1
1 2 2 2
Izlaz:
26.57
```

Zadatak 3.10 Napisati program u kome se prvo inicijalizuje statički niz struktura osoba sa članovima ime i prezime (uređen u rastućem poretku prezimena) sa manje od 10 elemenata, a zatim se učitava jedan karakter i pronalazi (bibliotečkom funkcijom `bsearch`) i štampa jedna struktura iz niza osoba čije prezime počinje tim karakterom. Ako takva osoba ne postoji, štampati -1 na standardni izlaz.

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

Test 1

```
Ulaz:  R
Izlaz: Dusko Radovic
```

3.2 Sortiranje

Zadatak 3.11 U datom nizu brojeva pronaći dva broja koja su na najmanjem rastojanju. Niz se zadaje sa standardnog ulaza, sve do kraja ulaza, i neće sadržati

više od 256 elemenata. Na izlaz ispisati njihovu razliku. Uputstvo: prvo sortirati niz.

Test 1

```
Ulaz:   23 64 123 76 22 7
Izlaz:  1
```

Test 2

```
Ulaz:   21 654 65 123 65 12 61
Izlaz:  0
```

Zadatak 3.12 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.

Test 1

```
Ulaz:   anagram ramgana
Izlaz:  jesu
```

Test 2

```
Ulaz:   anagram anagrm
Izlaz:  nisu
```

Zadatak 3.13 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 elemenata. Uputstvo: prvo sortirati niz, a zatim naći najdužu sekvencu jednakih elemenata.

Test 1

```
Ulaz:   4 23 5 2 4 6 7 34 6 4 5
Izlaz:  4
```

Test 2

```
Ulaz:   2 4 6 2 6 7 99 1
Izlaz:  2
```

Zadatak 3.14 Napisati funkciju koja proverava da li u datom nizu postoje dva elementa kojima je zbir zadati ceo broj. Napisati i program koji testira ovu funkciju. U programu se prvo učitava broj, a zatim i niz (pretpostaviti da za niz neće biti uneto više od 256 brojeva). Elementi niza se unose sve do kraja ulaza. Uputstvo: prvo sortirati niz.

Test 1

```
Ulaz:   34 134 4 1 6 30 23
Izlaz:  da
```

Test 2

```
Ulaz:   12 53 1 43 3 56 13
Izlaz:  ne
```

Zadatak 3.15 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 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 i program koji testira funkciju, u kome se nizovi unose sa standardnog ulaza, sve dok se ne unese 0. Dimenzija nizova neće biti preko 256.

Test 1

```
Ulaz:   3 6 7 11 14 35 0 3 5 8
   0
Izlaz:  3 3 5 6 7 8 11 14 35
```

Test 2

```
Ulaz:   1 4 7 0 9 11 23 54 75 0
Izlaz:  1 4 7 9 11 23 54 75
```

Zadatak 3.16 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 i kreira jedinstven spisak studenata sortiranih takođe po imenu rastuće. Program dobija nazive datoteka iz komandne linije, i jedinstven spisak upisuje u datoteku `ceo-tok.txt`. Pretpostaviti da je ime studenta nije duže od 10, a prezime od 15 karaktera.

Test 1

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

prvi-deo.txt:          drugi-deo.txt:          ceo-tok.txt:

Andrija Petrovic       Aleksandra Cvetic       (TODO)
Anja Ilic              Bojan Golubovic
Ivana Markovic         Dragan Markovic
Lazar Micic            Filip Dukic
Nenad Brankovic        Ivana Stankovic
Sofija Filipovic       Marija Stankovic
Vladimir Savic         Ognjen Peric
                       Uros Milic
```

Zadatak 3.17 Napraviti biblioteku `sort.h` i `sort.c` koja implementira algoritme sortiranja nizova celih brojeva. Biblioteka treba da sadrži `selection`, `merge`, `quick`, `bubble`, `insertion` i `shell sort`. Upotrebiti biblioteku kako bi se napravilo poređenje efikasnosti različitih algoritama sortiranja. Efikasnost meriti na slučajno generisanim nizovima, na već sortiranim nizovima i na naopako sortiranim nizovima. Izbor algoritma, veličine i početnog rasporeda elemenata niza birati kroz argumente komandne linije. Vreme meriti programom `time`. Analizirati porast vremena sa porastom dimenzije `n`.

Zadatak 3.18 Napisati funkcije koje sortiraju niz struktura tačaka na osnovu sledećih kriterijuma:

(a) njihovog rastojanja od koordinatnog početka,

(b) `x` koordinata tačaka,

(c) `y` koordinata tačaka.

Napisati program koji učitava niz tačaka iz datoteke čije se ime zadaje kao argument komandne linije, i u zavisnosti od prisutnih opcija u komandnoj liniji (`-d`, `-x` ili `-y`), sortira tačke po jednom od prethodna tri kriterijuma i rezultat upisuje u datoteku čije se ime zadaje kao drugi argument komandne linije. U ulaznoj datoteci nije zadato više od 128 tačaka.

Test 1

```
Poziv:  a.out -x tacke.txt
   sorttacke.txt
Ulazna datoteka:
3 4
11 6
7 3
2 82
-1 6
Izlazna datoteka:
-1 6
2 82
3 4
7 3
11 6
```

Zadatak 3.19 Napisati program koji učitava imena i prezimena građana (najviše njih 1000) 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 je za ime, odnosno prezime građana dovoljno 15 karaktera.

Test 1

```
biracki-spisak.txt:      Izlaz:
Andrija Petrovic         (TODO)
Anja Ilic
Aleksandra Cvetic
Bojan Golubovic
Dragan Markovic
Filip Dukic
Ivana Stankovic
Ivana Markovic
Lazar Micic
Marija Stankovic
```

Zadatak 3.20 Definisana je struktura podataka

```
typedef struct dete
{
      char ime[MAX_IME];
```

```
    char prezime[MAX_IME];
    unsigned godiste;
} Dete;
```

Napisati funkciju koja sortira niz dece po godištu, a kada su deca istog godišta, tada ih 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.

Test 1

```
Poziv: ./a.out ulaz.txt izlaz.txt
Ulazna datoteka:                Izlazna datoteka:
Petar Petrovic 2007             Marija Antic 2007
Milica Antonic 2008             Ana Petrovic 2007
Ana Petrovic 2007               Petar Petrovic 2007
Ivana Ivanovic 2009             Milica Antonic 2008
Dragana Markovic 2010           Ivana Ivanovic 2009
Marija Antic 2007               Dragana Markovic 2010
```

Zadatak 3.21 Napisati funkciju koja sortira niz niski po broju suglasnika u niski, ukoliko reči imaju isti broj suglasnika tada po dužini niske, a ukoliko su i dužine jednake onda leksikografski. 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 32 karaktera.

Test 1

```
Ulazna datoteka:
ana petar andjela milos nikola aleksandar ljubica matej milica
Izlaz:
ana matej milos petar milica nikola andjela ljubica aleksandar
```

Zadatak 3.22 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, prozivođačima i cenama učitati iz datoteke `artikli.txt`. Pretraživanje niza artikala vršiti binarnom pretragom.

Test 1

```
artikli.txt:                    Ulaz:           Izlaz:
1001 Keks Jaffa 120             (TODO)          (TODO)
2530 Napolitanke Bambi  230
0023 Medeno_srce Pionir 150
2145 Pardon Marbo 70
```

Zadatak 3.23 Napisati program koji iz datoteke `aktivnost.txt` čita podatke o

aktivnosti studenata na praktikumima i u datoteke `dat1.txt`, `dat2.txt` i `dat3.txt` upisuje redom tri spiska. Na prvom su studenti sortirani leksikografski po imenu rastuće. Na drugom su sortirani po ukupnom broju urađenih zadataka opadajuće, a ukoliko neki studenti imaju isti broj rešenih zadataka sortiraju 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 sortirati po prezimenu opadajuće. U datoteci se nalazi ime, prezime studenta, broj časova na kojima je prisustvovao, kao i ukupan broj urađenih 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:
   Izlaz:
Aleksandra Cvetic 4 6          (
   TODO)
Bojan Golubovic 4 3
Dragan Markovic 3 5
Filip Dukic 2 0
Ivana Stankovic 3 1
Marija Stankovic 1 3
Ognjen Peric 1 2
Uros Milic 2 5
Andrija Petrovic 2 5
Anja Ilic 3 1
Ivana Markovic 2 5
Lazar Micic 1 3
Nenad Brankovic 2 4
```

**** Zadatak 3.24** 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. Napomena: brojeve čuvati u sortiranom nizu i svaki naredni element umetati na svoje mesto. Optimizovati program, ukoliko se zna da neće biti više od 500 različitih unetih brojeva.

Test 1

```
Ulaz: U 2 U 0 U 6 U 4 N 1 U 8 N 2 N 5 U 2 N 3 N 5
Izlaz: 0 2 8 2 6
```

**** Zadatak 3.25** 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, sledeća slika 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 po-

tez 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.

```
3    5    2    1
4    4    1__  2
5__  3    3    3
1    1    4    4
2    2__  5    5
```

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.

3.3 Bibliotečke funkcije pretrage i sortiranja

Zadatak 3.26 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 i na standardni izlaz ispisati odgovarajuću poruku.

Test 1

```
TODO
```

Zadatak 3.27 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.

Test 1

```
TODO
```

Zadatak 3.28 Korišćenjem bibliotečke funkcije `qsort` napisati program koji sortira niz niski po sledećim kriterijumima:

(a) leksikografski,

(b) po dužini.

Niske se učitavaju iz fajla `niske.txt`, neće ih biti više od 1000, i svaka će biti dužine najviše 30 karaktera. Program prvo leksikografski sortira niz, primenjuje binarnu pretragu (`bsearch`) zarad traženja niske unete sa standardnog ulaza, a

potom linearnu pretragu koristeći funkciju `lfind`. Na kraju, niske bivaju sortirane po dužini. Rezultate svih sortiranja i pretraga ispisati na standardni izlaz.

Test 1

```
TODO
```

Zadatak 3.29 Uraditi prethodni zadatak 3.28 sa dinamički alociranim niskama, i sortiranjem niza pokazivača (umesto niza niski).

Test 1

```
TODO
```

Zadatak 3.30 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 broker bodova, ili poruka ukoliko nema takvog. Potom, sa standardnom 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 vise od 500 studenata, i da je za ime i prezime svakog studenta dovoljno po 20 karaktera.

Test 1

```
TODO
```

Zadatak 3.31 Uraditi zadatak 3.12, ali korišćenjem bibliotečke `qsort` funkcije.

Test 1

```
Ulaz:   vrata vatra
Izlaz:  jesu
```

Test 2

```
Ulaz:   qsort bsearch
Izlaz:  nisu
```

Zadatak 3.32 Napisati program koji sa standardnog ulaza učitava prvo ceo broj `n` ($n \leq 10$), a zatim niz `S` od `n` stringova (maksimalna dužina stringa je 32 karaktera). Sortirati niz `S` (bibliotečkom funkcijom `qsort`) i proveriti da li u njemu ima identičnih stringova.

Test 1

```
Ulaz:   4 prog search sort
   search
Izlaz:  ima
```

Test 2

```
Ulaz:   3 test kol ispit
Izlaz:  nema
```

Zadatak 3.33 Datoteka `studenti.txt` sadrži spisak studenata. Za svakog studenta poznat je nalog na Alas-u (oblika npr. `mr97125`, `mm09001`), ime i prezime i broj poena. Napisati program koji sortira (korišćenjem funkcije `qsort`) studente po broju poena (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. Ukoliko je u komandnoj liniji uz opciju `-n` naveden i nalog nekog studenta, funkcijom `bsearch` potražiti i prijaviti broj poena studenta sa tim nalogom. Sortirane studente upisati u datoteku `izlaz.txt`.

Test 1

```
Poziv: ./a.out -n mm13321
Datoteka:                          Izlaz:
mr14123 Marko Antic 20             mm13321 Marija Radic 12
mm13321 Marija Radic 12
ml13011 Ivana Mitrovic 19
ml13066 Pera Simic 15
mv14003 Jovan Jovanovic 17
```

Zadatak 3.34 Definisana je struktura:

```
typedef struct { int dan; int mesec; int godina; } Datum;}
```

Napisati funkciju koja poredi dva datuma 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 potom pozivanjem funkcije `bsearch` iz standardne biblioteke proverava da li datumi učitani sa standardnog ulaza (sve do kraja ulaza) postoje među prethodno unetim datumima.

Test 1

```
Poziv: ./a.out datoteka.txt
Datoteka:          Ulaz:            Izlaz:
1.1.2013           13.12.2016       postoji
13.12.2016         10.5.2015        ne postoji
11.11.2011         5.2.2009         postoji
3.5.2015
5.2.2009
```

3.4 Rešenja

Rešenje 3.1

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX 1000000
```

```
5
/* pri prevodjenju program linkovati sa bibliotekom librt opciom
7    -lrt zbog funkcije clock_gettime() */

9 /* Funkcija pretrazuje niz celih brojeva duzine n, trazeci u
     njemu element x. Pretraga se vrsi prostom iteracijom kroz
11   niz. Ako se element pronadje funkcija vraca indeks pozicije
     na kojoj je pronadjen. Ovaj indeks je uvek nenegativan. Ako
13   element nije pronadjen u nizu, funkcija vraca -1, kao
     indikator neuspesne pretrage. */
15 int linearna_pretraga(int a[], int n, int x)
{
17   int i;
   for (i = 0; i < n; i++)
19     if (a[i] == x)
       return i;
21   return -1;
}
23
/* Funkcija trazi u sortiranom nizu a[] duzine n broj x. Vraca
25   indeks pozicije nadjenog elementa ili -1, ako element nije
     pronadjen */
27 int binarna_pretraga(int a[], int n, int x)
{
29   int levi = 0;
   int desni = n - 1;
31   int srednji;
   /* Dokle god je indeks levi levo od indeksa desni */
33   while (levi <= desni) {
     /* Racunamo srednji indeks */
35     srednji = (levi + desni) / 2;
     /* Ako je srednji element veci od x, tada se x mora nalaziti
37        u levoj polovini niza */
     if (x < a[srednji])
39       desni = srednji - 1;
     /* Ako je srednji element manji od x, tada se x mora
41        nalaziti u desnoj polovini niza */
     else if (x > a[srednji])
43       levi = srednji + 1;
     else
45       /* Ako je srednji element jednak x, tada smo pronasli x na
            poziciji srednji */
47       return srednji;
   }
49   /* ako nije pronadjen vracamo -1 */
   return -1;
51 }

53 /* Funkcija trazi u sortiranom nizu a[] duzine n broj x. Vraca
     indeks pozicije nadjenog elementa ili -1, ako element nije
55   pronadjen */
int interpolaciona_pretraga(int a[], int n, int x)
57 {
   int levi = 0;
59   int desni = n - 1;
   int srednji;
```

```
61    /* Dokle god je indeks levi levo od indeksa desni...  */
      while (levi <= desni) {
63      /* Ako je element manji od pocetnog ili veci od poslednjeg
           clana u delu niza a[levi],...,a[desni] tada nije u tom
65         delu niza. Ova provera je neophodna, da se ne bi dogodilo
           da se prilikom izracunavanja indeksa srednji izadje izvan
67         opsega indeksa [levi,desni] */
        if (x < a[levi] || x > a[desni])
69        return -1;
        /* U suprotnom, x je izmedju a[levi] i a[desni], pa ako su
71         a[levi] i a[desni] jednaki, tada je jasno da je x jednako
           ovim vrednostima, pa vracamo indeks levi (ili indeks
73         desni, sve jedno je).Ova provera je neophodna, zato sto
           bismo inace prilikom izracunavanja srednji imali deljenje
75         nulom. */
        else if (a[levi] == a[desni])
77        return levi;
        /* Racunamo srednji indeks */
79      srednji =
            levi +
81          ((double) (x - a[levi]) / (a[desni] - a[levi])) *
            (desni - levi);
83      /* NAPOMENA: Indeks srednji je uvek izmedju levi i desni,
           ali ce verovatno biti blize trazenoj vrednosti nego da
85         smo prosto uvek uzimali srednji element. Ovo se moze
           porediti sa pretragom recnika: ako neko trazi rec na
87         slovo 'B', sigurno nece da otvori recnik na polovini, vec
           verovatno negde blize pocetku. */
89      /* Ako je srednji element veci od x, tada se x mora nalaziti
           u levoj polovini niza */
91      if (x < a[srednji])
          desni = srednji - 1;
93      /* Ako je srednji element manji od x, tada se x mora
           nalaziti u desnoj polovini niza */
95      else if (x > a[srednji])
          levi = srednji + 1;
97      else
          /* Ako je srednji element jednak x, tada smo pronasli x na
99           poziciji srednji */
          return srednji;
101   }
      /* ako nije pronadjen vracamo -1 */
103   return -1;
    }
105
    /* Funkcija main */
107 int main(int argc, char **argv)
    {
109   int a[MAX];
      int n, i, x;
111   struct timespec time1, time2, time3, time4, time5, time6;
      FILE *f;
113
      /* provera argumenata komandne linije */
115   if (argc != 3) {
        fprintf(stderr,
```

```
117             "koriscenje programa: %s dim_niza trazeni_br\n",
                argv[0]);
119     exit(EXIT_FAILURE);
    }
121
    /* Dimenzija niza */
123 n = atoi(argv[1]);
    if (n > MAX || n <= 0) {
125   fprintf(stderr, "Dimenzija niza neodgovarajuca\n");
      exit(EXIT_FAILURE);
127 }

129 /* Broj koji se trazi */
    x = atoi(argv[2]);
131
    /* Elemente niza odredjujemo slucajno, tako da je svaki
133    sledeci veci od prethodnog. srandom() funkcija obezbedjuje
       novi seed za pozivanje random() funkcije. Kako nas niz ne
135    bi uvek isto izgledao seed smo postavili na tekuce vreme u
       sekundama od Nove godine 1970. random()%100 daje brojeve
137    izmedju 0 i 99 */
    srandom(time(NULL));
139 for (i = 0; i < n; i++)
      a[i] = i == 0 ? random() % 100 : a[i - 1] + random() % 100;
141
    /* Lineara pretraga */
143 printf("Linearna pretraga\n");
    /* Racunamo vreme proteklo od Nove godine 1970 */
145 clock_gettime(CLOCK_REALTIME, &time1);
    /* Pretrazujemo niz */
147 i = linearna_pretraga(a, n, x);
    /* Racunamo novo vreme i razlika predstavlja vreme utroseno za
149    lin pretragu */
    clock_gettime(CLOCK_REALTIME, &time2);
151 if (i == -1)
      printf("Element nije u nizu\n");
153 else
      printf("Element je u nizu na poziciji %d\n", i);
155 printf("---------------------------------\n");

157 /* Binarna pretraga */
    printf("Binarna pretraga\n");
159 clock_gettime(CLOCK_REALTIME, &time3);
    i = binarna_pretraga(a, n, x);
161 clock_gettime(CLOCK_REALTIME, &time4);
    if (i == -1)
163   printf("Element nije u nizu\n");
    else
165   printf("Element je u nizu na poziciji %d\n", i);
    printf("---------------------------------\n");
167
    /* Interpolaciona pretraga */
169 printf("Interpolaciona pretraga\n");
    clock_gettime(CLOCK_REALTIME, &time5);
171 i = interpolaciona_pretraga(a, n, x);
    clock_gettime(CLOCK_REALTIME, &time6);
```

```
173   if (i == -1)
        printf("Element nije u nizu\n");
175   else
        printf("Element je u nizu na poziciji %d\n", i);
177   printf("---------------------------------\n");

179   /* Upisujemo podatke o izvrsavanju programa u log fajl */
      if ((f = fopen("vremena.txt", "a")) == NULL) {
181     fprintf(stderr, "Neuspesno otvaranje log fajla.\n");
        exit(EXIT_FAILURE);
183   }

185   fprintf(f, "Dimenzija niza od %d elemenata.\n", n);
      fprintf(f, "\tLinearna pretraga:%10ld ns\n",
187           (time2.tv_sec - time1.tv_sec) * 1000000000 +
              time2.tv_nsec - time1.tv_nsec);
189   fprintf(f, "\tBinarna: %19ld ns\n",
              (time4.tv_sec - time3.tv_sec) * 1000000000 +
191           time4.tv_nsec - time3.tv_nsec);
      fprintf(f, "\tInterpolaciona: %12ld ns\n\n",
193           (time6.tv_sec - time5.tv_sec) * 1000000000 +
              time6.tv_nsec - time5.tv_nsec);
195

      fclose(f);
197

      return 0;
199 }
```

Rešenje 3.2

```
#include <stdio.h>
2
int lin_pretgraga_rek(int a[], int n, int x)
4 {
    int tmp;
6   /* izlaz iz rekurzije */
    if (n <= 0)
8     return -1;
    /* ako je prvi element trazeni */
10  if (a[0] == x)                /* if (a[n-1] == x) */
      return 0;                   /* return n - 1; */
12  /* pretraga ostatka niza */
    tmp = lin_pretgraga_rek(a + 1, n - 1, x);
14  return tmp < 0 ? tmp : tmp + 1;
    /* return lin_pretgraga_rek(a, n - 1, x); */
16 }

18 int bin_pretgraga_rek(int a[], int l, int d, int x)
   {
20   int srednji;
     if (l > d)
22     return -1;
     /* srednja pozicija na kojoj se trazi vrednost x */
24   srednji = (l + d) / 2;
     /* ako je sredisnji element trazeni */
```

```
26    if (a[srednji] == x)
        return srednji;
28    /* ako je trazeni broj veci od srednjeg, pretrazujemo desnu
         polovinu niza */
30    if (a[srednji] < x)
        return bin_pretgraga_rek(a, srednji + 1, d, x);
32    /* ako je trazeni broj manji od srednjeg, pretrazujemo levu
         polovinu niza */
34    else
        return bin_pretgraga_rek(a, l, srednji - 1, x);
36  }

38
    int interp_pretgraga_rek(int a[], int l, int d, int x)
40  {
      int p;
42    if (x < a[l] || x > a[d])
        return -1;
44    if (a[d] == a[l])
        return l;
46    /* pozicija na kojoj se trazi vrednost x */
      p = l + (d - l) * (x - a[l]) / (a[d] - a[l]);
48    if (a[p] == x)
        return p;
50    if (a[p] < x)
        return interp_pretgraga_rek(a, p + 1, d, x);
52    else
        return interp_pretgraga_rek(a, l, p - 1, x);
54  }

56  #define MAX 1024

58  int main()
    {
60    int a[MAX];
      int x;
62    int i, indeks;

64    /* ucitavamo trazeni broj */
      scanf("%d", &x);
66
      /* ucitavamo elemente niza sve do kraja ulaza - ocekujemo da
68       korisnik pritisne CTRL+D za naznaku kraja */
      i = 0;
70    while (scanf("%d", &a[i]) == 1) {
        i++;
72    }

74    printf("Linearna pretraga\n");
      indeks = lin_pretgraga_rek(a, i, x);
76    if (indeks == -1)
        printf("Element se ne nalazi u nizu.\n");
78    else
        printf("Pozicija elementa je %d.\n", indeks);
80
      printf("Binarna pretraga\n");
```

```
82    indeks = bin_pretgraga_rek(a, 0, i - 1, x);
      if (indeks == -1)
84      printf("Element se ne nalazi u nizu.\n");
      else
86      printf("Pozicija elementa je %d.\n", indeks);

88    printf("Interpolaciona pretraga\n");
      indeks = interp_pretgraga_rek(a, 0, i - 1, x);
90    if (indeks == -1)
        printf("Element se ne nalazi u nizu.\n");
92    else
        printf("Pozicija elementa je %d.\n", indeks);
94
      return 0;
96  }
```

Rešenje 3.3

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

8 /* O svakom studentu imamo 3 informacije i njih objedinjujemo u
     strukturu kojom cemo predstavljati svakog studenta. */
10 typedef struct {
    /* indeks mora biti tipa long jer su podaci u datoteci
12     preveliki za int, npr. 20140123 */
    long indeks;
14  char ime[MAX_DUZINA];
    char prezime[MAX_DUZINA];
16 } Student;

18 /* Ucitan niz studenata ce biti sortiran prema indeksu, jer cemo
     ih, redom, kako citamo smestati u niz, a u datoteci su vec
20   smesteni sortirani rastuce prema broju indeksa. Iz tog
     razloga pretragu po indeksu cemo vrsiti binarnom pretragom,
22   dok pretragu po prezimenu moramo vrsiti linearno, jer nemamo
     garancije da postoji uredjenje po prezimenu. */
24
  /* Funkcija trazi u sortiranom nizu a[] duzine n broj x. Vraca
26   indeks pozicije nadjenog elementa ili -1, ako element nije
     pronadjen */
28 int binarna_pretraga(Student a[], int n, long x)
  {
30   int levi = 0;
    int desni = n - 1;
32  int srednji;
    /* Dokle god je indeks levi levo od indeksa desni */
34  while (levi <= desni) {
      /* Racunamo srednji indeks */
36    srednji = (levi + desni) / 2;
      /* Ako je srednji element veci od x, tada se x mora nalaziti
```

```
38          u levoj polovini niza */
      if (x < a[srednji].indeks)
40        desni = srednji - 1;
      /* Ako je srednji element manji od x, tada se x mora
42          nalaziti u desnoj polovini niza */
      else if (x > a[srednji].indeks)
44        levi = srednji + 1;
      else
46        /* Ako je srednji element jednak x, tada smo pronasli x na
             poziciji srednji */
48        return srednji;
    }
50  /* ako nije pronadjen vracamo -1 */
    return -1;
52  }

54  int linearna_pretraga(Student a[], int n, char x[])
    {
56    int i;
      for (i = 0; i < n; i++)
58      /* poredimo prezime i-tog studenta i poslato x */
        if (strcmp(a[i].prezime, x) == 0)
60        return i;
    return -1;
62  }

64  /* Main funkcija mora imate argumente jer se ime datoteke dobija
       kao argument komandne linije */
66  int main(int argc, char *argv[])
    {
68    /* Ucitacemo redom sve studente iz datoteke u niz. */
      Student dosije[MAX_STUDENATA];
70    FILE *fin = NULL;
      int i;
72    int br_studenata = 0;
      long trazen_indeks = 0;
74    char trazeno_prezime[MAX_DUZINA];

76    /* Proveravamo da li nam je korisnik prilikom poziva prosledio
         ime_datoteke sa informacijama o studentima */
78    if (argc != 2) {
        fprintf(stderr,
80              "Greska: Program se poziva sa %s ime_datoteke\n",
                argv[0]);
82      exit(EXIT_FAILURE);
      }
84
      /* Otvaramo datoteku */
86    fin = fopen(argv[1], "r");
      if (fin == NULL) {
88      fprintf(stderr,
              "Neuspesno otvaranje datoteke %s za citanje\n",
90            argv[1]);
        exit(EXIT_FAILURE);
92    }
```

```
94    /* Citamo sve dok imamo red sa informacijama o studentu */
      i = 0;
96    while (1) {
        if (i == MAX_STUDENATA)
98        break;
        if (fscanf
100         (fin, "%ld %s %s", &dosije[i].indeks, dosije[i].ime,
             dosije[i].prezime) != 3)
102       break;
        i++;
104   }
      br_studenata = i;
106
      /* Nakon citanja datoteka nam vise nije neophodna i odmah je
108      zatvaramo */
      fclose(fin);
110
      printf("Unesite indeks studenta cije informacije zelite: ");
112   scanf("%ld", &trazen_indeks);
      i = binarna_pretraga(dosije, br_studenata, trazen_indeks);
114   if (i == -1)
        printf("Ne postoji student sa indeksom %ld\n",
116             trazen_indeks);
      else
118     printf("Indeks: %ld, Ime i prezime: %s %s\n",
              dosije[i].indeks, dosije[i].ime, dosije[i].prezime);
120
      printf("Unesite prezime studenta cije informacije zelite: ");
122   scanf("%s", trazeno_prezime);
      i = linearna_pretraga(dosije, br_studenata, trazeno_prezime);
124   if (i == -1)
        printf("Ne postoji student sa prezimenom %s\n",
126             trazeno_prezime);
      else
128     printf("Indeks: %ld, Ime i prezime: %s %s\n",
              dosije[i].indeks, dosije[i].ime, dosije[i].prezime);
130
      return 0;
132 }
```

Rešenje 3.5

Rešenje 3.6

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

4 /* tacnost */
  #define EPS 0.001
6
  int main()
8 {
    double l, d, s;
10
```

```
  /* posto je u pitanju interval [0, 2] leva granica je 0, a
     desna 2 */
  l = 0;
  d = 2;

  /* sve dok ne pronadjemo trazenu vrednost argumenta */
  while (1) {
    /* pronalazimo sredinu intervala */
    s = (l + d) / 2;
    /* ako je vrednost kosinusa u ovoj tacki manja od zadate
       tacnosti, prekidamo pretragu */
    if (fabs(cos(s)) < EPS) {
      break;
    }
    /* ako je nula u levom delu intervala, nastavljamo pretragu
       na intervalu [l, s] */
    if (cos(l) * cos(s) < 0)
      d = s;
    else
      /* inace, nastavljamo pretragu na intervalu [s, d] */
      l = s;
  }

  /* stampamo vrednost trazene tacke */
  printf("%g\n", s);

  return 0;
}
```

Rešenje 3.11

```
#include<stdio.h>

int main(){
  printf("Hello bitovi!\n");
  return 0;
}
```

Glava 4

Dinamičke strukture podataka

4.1 Liste

Zadatak 4.1 Napisati program koji koristi jednostruko povezanu listu za čuvanje elemenata koji se unose sa standardnog ulaza. Unošenje novih brojeva u listu prekida se učitavanjem kraja ulaza (EOF). Svako dodavanje novog broja u listu ispratiti ispisivanjem trenutnog sadržaja liste.

(a) Definisati strukturu `Cvor` koja predstavlja čvor liste.

(b) Milena:Da li ovde treba dodati i funkciju koja kreira cvor? Nemam resenje kod sebe pa ne znam kako to vec ide, ali mislim da bi trebalo

(c) Napisati funkciju koja dodaje novi elemenat na početak liste.

(d) Napisati funkciju koja dodaje novi elemenat na kraj liste.

(e) Milena: Da li bi ovo trebalo izdvojiti u poseban zadatak? Nekako, ako dodajemo na pocetak i kraj, nemamo garanciju sortiranosti liste, tako da mi to nekako deluje da smo dva zadatka strpali u jedan. Napisati funkciju koja dodaje novi elemenat u listu tako da lista ostane rastuće sortirana.

(f) Napisati funkciju koja oslobađa memoriju koju je zauzela lista.

(g) Milena: Ova funkcija bi mogla razlicito da se implementira sa pretpostavkom da je lista sortirana i da nije sortirana, i zato mi dodatno deluje da bi ta dva zadatka trebalo razdvojiti Napisati funkciju koja pretražuje listu za elementom koji ima vrednost koja je argument funkcije.

(h) Napisati funkciju koja briše sve elemente u listi koji imaju vrednost koja je argument funkcije.

(i) Milena: Da li ovde nedostaje funckija koja oslobadja celu memoriju?

Sve funckije za rad sa listom najpre implementirati iterativno, a zatim i rekurzivno.

Zadatak 4.2 Napisati program koji koristi dvostruko povezanu listu za čuvanje celih brojeva koji se unose sa standardnog ulaza. Unošenje novih brojeva u listu se prekida učitavanjem kraja ulaza (EOF). Svako dodavanje novog broja u listu ispratiti ispisivanjem trenutnog sadržaja liste. I ovde isto mozda razdvojiti sortiranost od obične liste.

(a) Napisati funkciju koja dodaje novi elemenat na početak liste.

(b) Napisati funkciju koja dodaje novi elemenat na kraj liste.

(c) Napisati funkciju koja dodaje novi elemenat u listu tako da lista ostane rastuće sortirana.

(d) Napisati funkciju koja oslobađa memoriju koju je zauzela lista.

(e) Napisati funkciju koja pretražuje listu za elementom koji ima vrednost koja je argument funkcije.

(f) Napisati funkciju koja briše sve elemente u listi koji imaju vrednost koja je argument funkcije.

Sve funckije za rad sa listom implementirati iterativno.

Zadatak 4.3 Sadržaj datoteke je aritmetički izraz koji može sadržati zagrade {, [i (. Napisati program koji učitava sadržaj datoteke i korišćenjem steka utvrđuje da li su zagrade u aritmetičkom izrazu dobro uparene. Program štampa odgovarajuću poruku na standardni izlaz. Milena: promenjeni test primeri, voditi racuna u resenjima sta se stampa!

Test 1

```
Datoteka: {[23 + 5344] * (24 - 234)} - 23
Izlaz:  Zagrade su ispravno uparene.
```

Test 2

```
Datoteka: {[23 + 5] * (9 * 2)} - {23}
Izlaz:  Zagrade su ispravno uparene.
```

Test 3

```
Datoteka: {[2 + 54) / (24 * 87)} + (234 + 23)
Izlaz:  Zagrade nisu ispravno uparene.
```

Zadatak 4.4 Napisati program koji proverava ispravnost uparivanja etiketa u HTML datoteci. Ime datoteke se zadaje kao argument komandne linije . Milena: A sta ako se ne navede argument komandne linije? Uputstvo: za rešavanje problema koristiti stek implementiran preko listi čiji su čvorovi HTML etikete.

Test 1

```
Poziv: ./a.out datoteka.html
Datoteka.html:                              Izlaz:
<html>                                      Ispravno uparene etikete.
  <head><title>Primer</title></head>
  <body>
    <h1>Naslov</h1>
    Danas je lep i suncan dan. <br>
    A sutra ce biti jos lepsi.
    <a link="http://www.google.com"> Link 1</a>
    <a link="http://www.math.rs"> Link 2</a>
  </body>
</html>
```

Test 2

```
Poziv: ./a.out datoteka.html
Datoteka.html:                              Izlaz:
<html>                                      Neispravno uparene etikete.
  <head><title>Primer</title></head>
  <body>
</html>
```

Test 3

```
Poziv: ./a.out datoteka.html
Datoteka.html:                              Izlaz:
<html>                                      Neispravno uparene etikete.
  <head><title>Primer</title></head>
  <body>
  </body>
```

Zadatak 4.5 Milena: Problem sa ovim zadatkom je sto je program najpre na usluzi korisnicima, a zatim na usluzi sluzbeniku i to nekako zbunjuje u formulaicji. Formulacija mi nije bila jasna bez citanja resenja, pokusala sam da je preciziran, u nastavku je izmenjena formulacija.
Medjutim, ja i dalje nisam bas zadovoljna i zato predlazem da se formulacija izmeni tako da je program stalno na usluzi sluzbeniku. Program ucitva podatke o prijavljenim korisnicima iz datoteke. Sluzbenik odlucuje da li ce da obradjuje redom korisnike, ili ce u nekim situacijama da odlozi rad sa korisnikom i stavi ga na kraj reda. Program ga uvek pita da na osnovu jmbg-a i zahteva odluci da li ce ga staviti na kraj reda, ako hoce, on ide na kraj reda, ako nece, onda sluzbenik daje odgovor na zahtev i jmbg, zahtev i odgovor se upisuju u izlaznu datoteku.

Napisati program kojim se simulira rad jednog šaltera na kojem se prvo zakazuju termini, a potom službenik uslužuje korisnike redom, kako su se prijavljivali.

Korisnik se prijavljuje unošenjem svog `jmbg` broja (niska koja sadrži 13 karaktera) i zahteva (niska koja sadrži najviše 999 karaktera). Prijavljivanje korisnika se prekida

unošenjem karaktera za kraj ulaza (`EOF`).

Službenik redom proziva korisnike čitanjem njihovog `jmbg` broja, a zatim odlučuje da li korisnika vraća na kraj reda ili ga odmah uslužuje. Službeniku se postavlja pitanje `Da li korisnika vracate na kraj reda?` i ukoliko on da odgovor `Da`, korisnik se vraća na kraj reda. Ukoliko odgovor nije `Da`, tada službenik čita korisnikov zahtev. Posle svakog 10 usluženog korisnika, službeniku se nudi mogućnost da prekine sa radom, nevezano od broja korisnika koji i dalje čekaju u redu.

Za čuvanje korisničkih zahteva koristiti red implementiran korišćenjem listi.

Zadatak 4.6 Milena: Dodati sta se desava ako nije zadat argument komandne linije ili ako datoteka ne postoji Napisati program koji prebrojava pojavljivanja etiketa HTML datoteke čije se ime zadaje kao argument komandne linije. Rezultat prebrojavanja ispisati na standardni izlaz. Etikete smeštati u listu, a za formiranje liste koristiti strukturu:

```
typedef struct _Element
{
  unsigned broj_pojavljivanja;
  char etiketa[20];
  struct _Element *sledeci;
} Element;
```

Test 1

```
Poziv: ./a.out datoteka.html
Datoteka.html:                                          Izlaz:
<html>                                                  a - 4
  <head><title>Primer</title></head>                    br - 1
  <body>                                                h1 - 2
    <h1>Naslov</h1>                                     body - 2
    Danas je lep i suncan dan. <br>                     title - 2
    A sutra ce biti jos lepsi.                          head - 2
    <a link="http://www.google.com"> Link 1</a>         html - 2
    <a link="http://www.math.rs"> Link 2</a>
  </body>
</html>
```

Zadatak 4.7 Milena: i ovde dodati sta ako nema argumenata i ako nema datoteka, kao i u svim ostalim zadacima, a ne bih stalno ovaj komentar ponavljala. Takodje, malo me muci u ovom zadatku sto nema neki smisao. Naime, ako se samo vrsi ucitavanje iz datoteka i ispisivanje, onda su ove liste zapravo visak jer isti rezultat moze da se dobije i bez koriscenja listi. Zato mi fali da program uradi nesto sto ne bi mogao da uradi bez koriscenja listi, npr da na osnovu unetog broja ispisuje svaki n-ti broj rezultujuce liste pa to u nekoj petlji da korisnik moze da ispisuje za razlicite unete n ili tako nesto...

Napisati program koji objedinjuje dve sortirane liste. Funkcija ne treba da kreira nove čvorove, već da samo postojeće čvorove preraspodeli. Prva lista se učitava iz datoteke koja se zadaje kao prvi argument komandne linije, a druga iz datoteke čije

se ime zadaje kao drugi argument komandne linije. Rezultujuću listu ispisati na standardni izlaz.

Test 1

```
Poziv: ./a.out dat1.txt dat2.
    txt
dat1.txt: 2 4 6 10 15
dat2.txt: 5 6 11 12 14 16
Izlaz: 2 4 5 6 6 10 11 12 14 15
     16
```

Zadatak 4.8 Napisati funkciju koja formira listu studenata tako što se podaci o studentima učitavaju iz datoteke čije se ime zadaje kao argument komandne linije. U svakom redu datoteke nalaze se podaci o studentu i to broj indeksa, ime i prezime. Napisati rekurzivnu funkciju koja određuje da li neki student pripada listi ili ne. Ispisati zatim odgovarajuću poruku i rekurzivno osloboditi memoriju koju je data lista zauzimala. Student se traži na osnovu broja indeksa, koji se zadaje sa standardnog ulaza.

Test 1

```
Poziv: ./a.out studenti.txt
Datoteka:                Ulaz:       Izlaz:
123/2014 Marko Lukic     3/2014      da: Ana Sokic
3/2014 Ana Sokic         235/2008    ne
43/2013 Jelena Ilic      41/2009     da: Marija Zaric
41/2009 Marija Zaric
13/2010 Milovan Lazic
```

Milena: Imamo dva zadatka sa labelom 608!

Zadatak 4.9 Neka su date dve jednostruko povezane liste L1 i L2. Napisati funkciju koja od tih lista formira novu listu L koja sadrži alternirajući raspoređene elemente lista L1 i L2 (prvi element iz L1, prvi element iz L2, drugi element L1, drugi element L2, itd). Ne formirati nove čvorove, već samo postojeće čvorove rasporediti u jednu listu. Prva lista se učitava iz datoteke koja se zadaje kao prvi argument komandne linije, a druga iz datoteke čije se ime zadaje kao drugi argument komandne linije. Rezultujuću listu ispisati na standardni izlaz. Milena: Sta ako je neka lita duza? To precizirati. I ovde me muci sto nedostaje neki smisao zadatku, nesto sto ne bi moglo da se uradi da nismo kristili liste.

Test 1

```
Poziv: ./a.out dat1.txt dat2.
   txt
dat1.txt: 2 4 6 10 15
dat2.txt: 5 6 11 12 14 16
Izlaz:  2 5 4 6 6 11 10 12 15
   14 16
```

Zadatak 4.10 Data je datotka brojevi.txt koja sadrži cele brojeve, po jedan u svakom redu.

(a) Napisati funkciju koja iz zadate datoteke učitava brojeve i smešta ih u listu.

(b) Napisati funkciju koja u jednom prolazu kroz zadatu listu celih brojeva pronalazi maximalan strogo rastući podniz.

Napisati program koji u datoteku `Rezultat.txt` upisuje nađeni strogo rastući podniz. Milena: I ovde me muci sto bi zadatak mogao da se resi i bez koriscenja listi...

Milena: Prirodni oblik testa ovde bi bio horizontalan, a ne ovako vertikalan.

Test 1

```
Ulaz:  brojevi.txt        Izlaz: Rezultat.txt
       43                        12
       12                        15
       15                        16
       16
       4
       2
       8
```

Zadatak 4.11 Grupa od n plesača na kostimima imaju brojeve od 1 do n, redom, u smeru kazaljke na satu. Plesači izvode svoju plesnu tačku tako što formiraju krug iz kog najpre izlazi k-ti plesač. Odbrojava se počevši od plesača označenog brojem 1 u smeru kretanja kazaljke na satu. Preostali plesači obrazuju manji krug iz kog opet izlazi k-ti plesač. Odbrojavanje počinje od sledećeg suseda prethodno izbačenog, opet u smeru kazaljke na satu. Izlasci iz kruga se nastavljaju sve dok svi plesači ne budu isključeni. Celi brojevi n, k ($k < n$) se učitavaju sa standardnog ulaza. Napisati program koji će na standardni izlaz ispisati redne brojeve plesača u redosledu napuštanja kruga. Uputstvo: u implementaciji koristiti kružnu listu.

Test 1

```
Ulaz: 5 3
Izlaz: 3 1 5 2 4
```

Milena: Bilo bi lepo dodati i prethodni zadatak u kojem se smer izbacivanja stalno menja, tako da se onda korsti dvostruko povezana kružna lista.

4.2 Drveta

4.3 Rešenja

Rešenje 4.1

```
#include<stdio.h>

int main(){
  printf("Hello bitovi!\n");
  return 0;
}
```

Rešenje 4.2

```
#include<stdio.h>

int main(){
  printf("Hello bitovi!\n");
  return 0;
}
```

Rešenje 4.8

```
#include<stdio.h>

int main(){
  printf("Hello bitovi!\n");
  return 0;
}
```

Rešenje 4.4

```
#include<stdio.h>

int main(){
  printf("Hello bitovi!\n");
  return 0;
}
```

Rešenje 4.5

```
#include<stdio.h>

int main(){
```

```
4   printf("Hello bitovi!\n");
    return 0;
6 }
```

Rešenje 4.6

```
#include<stdio.h>
2
int main(){
4   printf("Hello bitovi!\n");
    return 0;
6 }
```

Rešenje 4.7

```
#include<stdio.h>
2
int main(){
4   printf("Hello bitovi!\n");
    return 0;
6 }
```

Rešenje 4.8

```
1 #include<stdio.h>

3 int main(){
    printf("Hello bitovi!\n");
5   return 0;
}
```

Rešenje 4.9

```
#include<stdio.h>
2
int main(){
4   printf("Hello bitovi!\n");
    return 0;
6 }
```

Rešenje 4.10

```
#include<stdio.h>
2
int main(){
4   printf("Hello bitovi!\n");
    return 0;
6 }
```

Rešenje 4.11

```
#include<stdio.h>

int main(){
  printf("Hello bitovi!\n");
  return 0;
}
```

Glava 5

Ispitni rokovi

5.1 Programiranje 2, praktični deo ispita, jun 2015.

Zadatak 5.1

Kao argument komandne linije zadaje se ime ulazne datoteke u kojoj se nalaze niske. U prvoj liniji datoteke nalazi se informacija o broju niski, a zatim u narednim linijama po jedna niska ne duža od `50` karaktera.

Napisati program u kojem se dinamički alocira memorija za zadati niz niski, a zatim se na standardnom izlazu u redosledu suprotnom od redosleda čitanja ispisuju sve niske koje počinju velikim slovom.

U slučaju pojave bilo kakve greške na standardnom izlazu ispisati vrednost `-1` i prekinuti izvršavanje programa.

Test 1

```
Sadržaj datoteke:
 5
 Programiranje
 Matematika
 12345
 dInAmiCnArEc
 Ispit
Izlaz:
 Ispit
 Matematika
 Programiranje
```

Test 2

```
Sadržaj datoteke:
  2
  maksimalano
  poena
Izlaz:
```

Test 3

```
Problem:
  datoteka
   ne postoji
Izlaz:
  -1
```

Zadatak 5.2

Data je biblioteka za rad sa binarnim pretrazivačkim stablima čiji čvorovi sadrže cele brojeve. Napisati funkciju `int sumirajN (Cvor * koren, int n)` koja izračunava zbir svih čvorova koji se nalaze na n-tom nivou stabla (koren se nalazi na nultom nivou, njegova deca na prvom nivou i tako redom). Ispravnost napisane funkcije testirati na osnovu zadate `main` funkcije i biblioteke za rad sa pretraživačkim stablima.

Napisati program koji sa standardnog ulaza učitava najpre prirodan broj n, a

potom i brojeve sve do pojave nule koje smešta u stablo i ispisuje rezultat pozivanja funkcije `prebrojN` za broj n i tako kreirano stablo. U slučaju greške na standardni izlaz za grešku ispisati `-1`.

Test 1

```
Ulaz:
 2 8 10 3 6 14 13 7 4 0
Izlaz:
 20
```

Test 2

```
Ulaz:
 0 50 14 5 2 4 56 8 52 7 1 0
Izlaz:
 50
```

Zadatak 5.3 Sa standardnog ulaza učitava se broj vrsta i broj kolona celobrojne matrice A, a zatim i elementi matrice A. Napisati program koji će ispisati indeks kolone u kojoj se nalazi najviše negativnih elemenata. Ukoliko postoji više takvih kolona, ispisati indeks prve kolone. Može se pretpostaviti da je broj vrsta i broj kolona manji od `50`. U slučaju greške ispisati vrednost `-1` na standardni izlaz za greške.

Test 1

```
Ulaz:
  4
  5
  1  2  3  4  5
 -1  2 -3  4 -5
 -5 -4 -3 -2  1
 -1  0  0  0  0
Izlaz:
 0
```

Test 2

```
Ulaz:
 2
 3
 0 0 -5
 1 2 -4
Izlaz:
```

Test 3

```
Ulaz:
 -2
Izlaz (na stderr):
 -1
```

5.2 Programiranje 2, praktični deo ispita, jul 2015.

Zadatak 5.4

Napisati program koji kao prvi arugment komandne linije prima ime dokumenta u kome treba prebrojati sva pojavljivanja tražene niske (bez preklapanja) koja se navodi kao drugi argument komandne linije (iskoristiti funkciju standardne biblioteke `strstr`). U slučaju bilo kakve greške ispisati `-1` na standardni izlaz za greške. Pretpostaviti da linije datoteke neće biti duže od `127` karaktera.
Potpis funkcije strstr:
`char *strstr(const char *haystack, const char *needle);`
Funkcija traži prvo pojavljivanje podniske needle u nisci haystack, i vraća pokazivač na početak podniske, ili NULL ako podniska nije pronađena.

Test 1

```
Poziv:  ./a.out fajl.txt test
Datoteka:   Ovo je test primer.
        U njemu se rec test
   javlja
        vise puta. testtesttest
Izlaz:  5
```

Test 2

```
Poziv:   ./a.out
Izlaz  (na stderr):  -1
```

Test 3

```
Poziv:   ./a.out fajl.txt foo
Datoteka:   (ne postoji)
Izlaz (na stderr):    -1
```

Test 4

```
Poziv:  ./a.out fajl.txt .
Datoteka:   (prazna)
Izlaz:    0
```

Zadatak 5.5

Na početku datoteke "trouglovi.txt" nalazi se broj trouglova čije su koordinate temena zapisane u nastavku datoteke. Napisati program koji učitva trouglove, i ispisuje ih na standardni izlaz sortirane po površini opadajuće (koristiti Heronov obrazac: $P = \sqrt{s * (s-a) * (s-b) * (s-c)}$, gde je s poluobim trougla). U slučaju bilo kakve greške ispisati `-1` na standardni izlaz za greške. Ne praviti nikave pretpostavke o broju trouglova u datoteci, i proveriti da li je datoteka ispravno zadata.

Test 1

```
Datoteka:  4
           0 0 0 1.2 1 0
           0.3 0.3 0.5 0.5 0.9
   1
           -2 0 0 0 0 1
           2 0 2 2 -1 -1
Izlaz:     2 0 2 2 -1 -1
           -2 0 0 0 0 1
           0 0 0 1.2 1 0
           0.3 0.3 0.5 0.5 0.9
   1
```

Test 2

```
Datoteka:  3
           1.2 3.2
   1.1 4.3
Izlaz:     -1
```

Test 3

```
Datoteka:   (nema datoteke)
Izlaz:     -1
```

Test 4

```
Datoteka:   0
Izlaz:
```

Zadatak 5.6 Data je biblioteka za rad sa binarnim pretraživačkim stablima celih brojeba. Napisati funkciju

`int f3(Cvor *koren, int n)`

koja u datom stablu prebrojava čvorove na n-tom nivou, koji imaju tačno jednog potomka. Pretpostaviti da se koren nalazi na nivou `0`. Ispravnost napisane funkcije testirati na osnovu zadate `main` funkcije i biblioteke za rad sa stablima.

Test 1

```
Ulaz:
 1 5 3 6 1 4 7 9
Izlaz:
 1
```

Test 2

```
Ulaz:
 2 5 3 6 1 0 4 7 9
Izlaz:
 2
```

Test 3

```
Ulaz:
 0 4 2 5
Izlaz:
 0
```

Test 4

```
Ulaz:
  3
Izlaz:
 0
```

Test 5

```
Ulaz:
  -1 4 5 1 7
Izlaz:
 0
```

5.3 Rešenja

Rešenje 5.1

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#define MAX 50

void greska(){
    printf("-1\n");
    exit(EXIT_FAILURE);
}

int main(int argc, char* argv[]){

    FILE* ulaz;
    char** linije;
    int i, j, n;

    /* Proveravamo argumente komandne linije.
  */
    if(argc!=2){
        greska();
    }

    /* Otvaramo datoteku čije ime je navedeno kao argument
    komandne linije neposredno nakon imena programa koji se
    poziva. */
    ulaz=fopen(argv[1], "r");
    if(ulaz==NULL){
        greska();
    }

    /* Učitavamo broj linija. */
    fscanf(ulaz, "%d", &n);

    /* Alociramo memoriju na osnovu učitanog broja linija.*/
```

```
33      linije=(char**)malloc(n*sizeof(char*));
        if(linije==NULL){
35          greska();
        }
37      for(i=0; i<n; i++){
            linije[i]=malloc(MAX*sizeof(char));
39          if(linije[i]==NULL){
                for(j=0; j<i; j++){
41                  free(linije[j]);
                }
43              free(linije);
                greska();
45          }
        }
47
        /* Učitavamo svih n linija iz datoteke. */
49      for(i=0; i<n; i++){
            fscanf(ulaz, "%s", linije[i]);
51      }

53      /* Ispisujemo u odgovarajućem poretku učitane linije koje
       zadovoljavaju kriterijum. */
        for(i=n-1; i>=0; i--){
55          if(isupper(linije[i][0])){
                printf("%s\n", linije[i]);
57          }
        }
59
        /* Oslobađamo memoriju koju smo dinamički alocirali. */
61      for(i=0; i<n; i++){
            free(linije[i]);
63      }

65      free(linije);

67      /* Zatvaramo datoteku. */
        fclose(ulaz);
69
        /* Završavamo sa programom. */
71      return 0;

73  }
```

Rešenje 5.2

```
#include <stdio.h>
2 #include "stabla.h"


4
int sumirajN (Cvor * koren, int n){
6       if(koren==NULL){
                return 0;
8       }

10      if(n==0){
```

```
        return koren->broj;
    }

    return sumirajN(koren->levo, n-1) + sumirajN(koren->desno, n-1);
}


int main(){
    Cvor* koren=NULL;
    int n;
    int nivo;

    /* Čitamo vrednost nivoa */
    scanf("%d", &nivo);


    while(1){

        /* Čitamo broj sa standardnog ulaza */
        scanf("%d", &n);

        /* Ukoliko je korisnik uneo 0, prekidamo dalje čitanje. */
        if(n==0){
            break;
        }

        /* A ako nije, dodajemo procitani broj u stablo. */
        dodaj_u_stablo(&koren, n);

    }

    /* Ispisujemo rezultat rada tražene funkcije */
    printf("%d\n", sumirajN(koren,nivo));

    /* Oslobađamo memoriju */
    oslobodi_stablo(&koren);


    /* Prekidamo izvršavanje programa */
    return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include "stabla.h"

Cvor* napravi_cvor(int b ) {
    Cvor* novi = (Cvor*) malloc(sizeof(Cvor));
    if( novi == NULL)
        return NULL;

    /* Inicijalizacija polja novog čvora */
    novi->broj = b;
    novi->levo = NULL;
```

```
13      novi->desno = NULL;

15      return novi;
}
17

19  void oslobodi_stablo(Cvor** adresa_korena)  {
        /* Prazno stablo i nema šta da se oslobađa */
21      if( *adresa_korena == NULL)
            return;
23
        /* Rekurzivno oslobađamo najpre levo, a onda i desno
       podstablo*/
25      if( (*adresa_korena)->levo )
            oslobodi_stablo(&(*adresa_korena)->levo);
27      if( (*adresa_korena)->desno)
            oslobodi_stablo(&(*adresa_korena)->desno);
29
        free(*adresa_korena);
31      *adresa_korena =NULL;
}
33

35  void proveri_alokaciju( Cvor* novi) {
        if( novi == NULL) {
37          fprintf(stderr, "Malloc greska za nov cvor!\n");
            exit(EXIT_FAILURE);
39      }
}
41
    void dodaj_u_stablo(Cvor** adresa_korena, int broj) {
43      /* Postojece stablo je prazno*/
        if( *adresa_korena == NULL){
45          Cvor* novi = napravi_cvor(broj);
            proveri_alokaciju(novi);
47          *adresa_korena = novi;  /*  Kreirani čvor novi će biti
       od sada koren stabla*/
            return;
49      }

51      /* Brojeve smeštamo u uređeno binarno stablo, pa
        ako je broj koji ubacujemo manji od broja koji je u korenu
       */
53      if( broj < (*adresa_korena)->broj)          /* dodajemo u
       levo podstablo */
            dodaj_u_stablo(&(*adresa_korena)->levo, broj);
55      /* ako je broj manji ili jednak od broja koji je u korenu
       stabla, dodajemo nov čvor desno od korena */
        else
57          dodaj_u_stablo(&(*adresa_korena)->desno, broj);
}
```

```
1   #ifndef __STABLA_H__
    #define __STABLA_H__ 1
3
    /* Struktura kojom se predstavlja čvor drveta */
```

```c
typedef struct dcvor{
    int broj;
    struct dcvor* levo, *desno;
} Cvor;

/* Funkcija alocira prostor za novi čvor drveta, inicijalizuje
   polja
    strukture i vraća pokazivač na nov čvor */
Cvor* napravi_cvor(int b );

/* Oslobađamo dinamički alociran prostor za stablo
 * Nakon oslobađanja se u pozivajućoj funkciji koren
 * postavljana NULL, jer je stablo prazno  */
void oslobodi_stablo(Cvor** adresa_korena);


/* Funkcija proverava da li je novi čvor ispravno alociran,
 * i nakon toga prekida program */
void proveri_alokaciju( Cvor* novi);


/* Funkcija dodaje nov čvor u stablo i
 * ažurira vrednost korena stabla u pozivajućoj funkciji.
 */
void dodaj_u_stablo(Cvor** adresa_korena, int broj);

#endif
```

Rešenje 5.3

```c
#include <stdio.h>
#define MAX 50


int main(){
    int m[MAX][MAX];
    int v, k;
    int i, j;
    int max_broj_negativnih, max_indeks_kolone;
    int broj_negativnih;

    /* Učitavamo dimenzije matrice */
    scanf("%d", &v);
    scanf("%d", &k);

    if(v<0 || v>MAX || k<0 || k>MAX){
        fprintf(stderr, "-1\n");
        return 0;
    }

    /* Učitavamo elemente matrice */
    for(i=0; i<v; i++){
        for(j=0; j<k; j++){
            scanf("%d", &m[i][j]);
        }
```

```
    }

    /*Pronalazimo kolonu koja sadrži najveći broj negativnih
    elemenata */
    max_indeks_kolone=0;

    max_broj_negativnih=0;
    for(i=0; i<v; i++){
        if(m[i][0]<0){
            max_broj_negativnih++;
        }

    }

    for(j=0; j<k; j++){
        broj_negativnih=0;
        for(i=0; i<v; i++){
            if(m[i][j]<0){
                broj_negativnih++;
            }
            if(broj_negativnih>max_broj_negativnih){
                max_indeks_kolone=j;
            }
        }

    }

    /* Ispisujemo traženi rezultat */
    printf("%d\n", max_indeks_kolone);

    /* Završavamo program */
    return 0;
}
```

Rešenje 5.4

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 128

int main(int argc, char **argv) {
  FILE *f;
  int brojac = 0;
  char linija[MAX], *p;

  if (argc != 3) {
    fprintf(stderr, "-1\n");
    exit(EXIT_FAILURE);
  }

  if ((f = fopen(argv[1], "r")) == NULL) {
    fprintf(stderr, "-1\n");
    exit(EXIT_FAILURE);
  }
```

```
21    while (fgets(linija, MAX, f) != NULL) {
        p = linija;
23      while (1) {
          p = strstr(p, argv[2]);
25        if (p == NULL)
    break;
27          brojac++;
          p = p + strlen(argv[2]);
29      }
    }
31
    fclose(f);
33
    printf("%d\n", brojac);
35
    return 0;
37  }
```

Rešenje 5.5

Rešenje 5.6