Priprema za II pismeni zadatak

Stringovi

1. Napisati program koji čita jednu reč teksta koju čine samo velika slova i ispisuje slovo koji se najčešće pojavljuje i koliko puta se pojavljuje. Ako se više slova najčešće pojavljuje, ispisuje se slovo koji se pre pojavilo u reči. U jednoj liniji standardnog ulaza nalazi se jedna reč teksta sa ne više od 20 slova. U prvoj liniji standarnog izlaza prikazati slovo koje se najčešće pojavljuje, a u drugoj liniji standardnog ulaza ispisati i koliko puta se pojavljuje.

ULAZ	IZLAZ
GREEEKKK	E 3
BAABC	B 2
ABCDE	A 1

Rešenje 01:

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;

int main() {
  // rec koja se analizira
  string rec;
  cin >> rec;

  // broj pojavljivanja svakog slova od A do Z
  int brojPojavljivanja[26] = {0};
  // uvecavamo broj pojavljivanja svakog slova iz reci
  for (int i = 0; i < rec.size(); i++)
    brojPojavljivanja[rec[i] - 'A']++;

  // odredjujemo najveci broj pojavljivanja slova koristeci max_element iz <algorithm>
  int maxPojavljivanja = *max_element(brojPojavljivanja,
                                      next(brojPojavljivanja, 26));


  // ispisujemo slovo koje se prvo pojavljuje u reci sa tim brojem
  // pojavljivanja
  for (int i = 0; i < rec.size(); i++)
    if (brojPojavljivanja[rec[i] - 'A'] == maxPojavljivanja) {
      cout << rec[i] << endl
           << maxPojavljivanja << endl;
      break;
```

```
    }

  return 0;
}
```

OBJAŠNJENJE

Ključni deo zadataka je da se za svako veliko slovo engleskog alfabeta izbroji koliko se puta pojavljuje u datoj reči.

Dakle, potrebno je uvesti 26 različitih brojača - po jedan za svako veliko slovo engleskog alfabeta. Jasno je da uvođenje 26 različitih promenljivih ne dolazi u obzir. Potrebna je struktura podataka koja preslikava dati karakter u broj njegovih pojavljivanja. Ovakve strukture podataka nazivaju se asocijativni nizovi, konačna preslikavanja, mape ili rečnici i one preslikavaju date ključeve (u našem slučaju ključevi su karakteri) u njima pridružene vrednosti (u našem slučaju to su brojevi pojavljivanja).

Najbolja takva struktura u ovom zadatku je niz brojača u kojem se na poziciji nula čuva broj pojavljivanja slova `A`, na poziciji jedan slova `B` itd., sve do pozicije 25 na kojoj se nalazi broj pojavljivanja slova`Z`.

Pitanje: Kako na osnovu karaktera odrediti poziciju njegovog brojača u nizu?
Odgovor: Koristimo činjenicu da su karakterima pridruženi numerički kodovi (ASCII u jeziku C++ tj. Unicode u jeziku Java i C#) i to na osnovu redosleda karaktera u engleskom alfabetu.
Redni broj karaktera je zato moguće odrediti oduzimanjem koda karaktera `A` od koda tog karaktera (na primer, kod karaktera `D` je 68, dok je kod karaktera `A` 65, oduzimanjem se dobija 3, što znači da se brojač karaktera `D` nalazi na mestu broj 3 u nizu).

Preslikavanje karaktera u njihov broj pojavljivanja moguće je ostvariti i bibliotečkim strukturama podataka koje predstavljaju tzv. Mape tj. rečnike (struktura podataka Dictionary u nekim programskim jezicima). U jeziku c++ bismo mogli upotrebiti `map<char, int>` ili `unordered_map<char, int>`
Ipak, ove strukture podataka su primerenije za situacije u kojima nije tako jednostavno na osnovu ključa dobiti njihovu numeričku vrednost i kada je skup dopustivih ključeva širi.

Rešenje 02:

```
#include <iostream>
#include<cstdio>
using namespace std;

int main()
{
   char rec[21];
   gets(rec);
   int brPojava[26];

   for(int i=0; i<26; i++)
      brPojava[i]=0;
```

```
    int maxPojava=0;
    /* 1. prolaz kroz niz */
    for(int i=0; rec[i]; i++)
    {
        brPojava[rec[i]-'A']++;
        if(brPojava[rec[i]-'A']>maxPojava)
            maxPojava=brPojava[rec[i]-'A'];
    }

    /* 2. prolaz kroz niz */
    for(int i=0; rec[i]; i++)
    {
        if(brPojava[rec[i]-'A']==maxPojava)
        {
            cout<<rec[i]<<endl<<maxPojava<<endl;
            break;
        }
    }
    return 0;
}
```

OBJAŠNJENJE

Zadatak možemo rešavati tako što ćemo dva puta proći kroz niz unetih slova tj. reč.
1. U prvom prolazu brojač pojavljivanja svakog velikog slova na koje naiđemo uvećavamo za 1. Nakon toga (ili paralelno sa tim) određujemo i maksimalni broj pojavljivanja nekog slova (za to koristimo uobičajene načine za određivanje maksimuma niza)
2. U drugom prolazu kroz niz unetih slova tj. reč tražimo prvo slovo reči čiji broj pojava je jednak već nađenom najvećem broju pojava nekog slova. Kada ga nađemo ispisujemo ga i prekidamo petlju (na primer, naredbom `break`).
Primetimo da ovde zapravo vršimo jednostavnu linearnu pretragu

Napomenimo i to da je za predstavljanje reči moguće upotrebiti ili tip `string` ili niz karaktera terminisan nulom (što je više u duhu jezika C).

Rešenje 03 (C++ 11, STL)

```
#include <iostream>
#include <string>
#include <algorithm>
#include <map>
using namespace std;

int main() {
  // rec koja se analizira
  string rec;
  cin >> rec;

  // broj pojavljivanja svakog slova od A do Z
  map<char, int> brojPojavljivanja;
```

```
  // uvecavamo broj pojavljivanja svakog slova iz reci
  for (int i = 0; i < rec.size(); i++)
    brojPojavljivanja[rec[i]]++;

  // odredjujemo najveci broj pojavljivanja slova
  int maxPojavljivanja = 0;
  for (auto it = brojPojavljivanja.begin(); it != brojPojavljivanja.end(); it++)
    if (it->second > maxPojavljivanja)
      maxPojavljivanja = it->second;

  // ispisujemo slovo koje se prvo pojavljuje u reci sa tim brojem
  // pojavljivanja
  for (int i = 0; i < rec.size(); i++)
    if (brojPojavljivanja[rec[i]] == maxPojavljivanja) {
      cout << rec[i] << endl
           << maxPojavljivanja << endl;
      break;
    }

  return 0;
}
```

2. Date su dve reči zapisane malim slovima. Napisati program kojim se proverava da li se druga reč može dobiti precrtavanjem slova (ne obavezno susednih) u prvoj reči. U prvoj liniji standardnog ulaza nalazi se prva reč, a u drugoj liniji druga reč. U prvoj liniji standardnog izlaza prikazati reč `da` ako se druga reč može dobiti precrtavanjem nekih slova prve reči, inače prikazati reč `ne`.

Primer 1
Ulaz
abeceda
abc

Izlaz
da

Primer 2
Ulaz
abeceda
cba

Izlaz
ne

Rešenje 01

```
#include <iostream>
#include <string>
using namespace std;

int main() {
  // ucitavamo dve reci
```

```
  string s1, s2;
  cin >> s1 >> s2;

  // redom prolazimo kroz slova obe reci dok ne dodjemo do kraja jedne
  // od njih
  int i = 0, j = 0;
  while (i < s1.size() && j < s2.size()) {
    // ako je tekuce slovo prve reci jednako tekucem slovu druge reci
    // onda ga zadrzavamo, a u suprotnom ga precrtavamo
    if (s1[i] == s2[j])
      // ako smo slovo zadrzali, prelazimo na naredno slovo druge reci
      j++;
    // prelazimo na naredno slovo prve reci
    i++;
  }

  // rec se moze dobiti ako i samo ako smo stigli do kraja druge reci
  // (sva slova druge reci smo pronasli u prvoj)
  if (j == s2.size())
    cout << "da" << endl;
  else
    cout << "ne" << endl;
  return 0;
}
```

OBJAŠNJENJE

При прецртавању слова, њихов редослед се не мења што значи да и у другој речи редослед слова мора бити исти као у првој. То значи да је довољно пролазити у циклусу кроз прву реч, карактер по карактер, и проверавати да ли је текући карактер једнак карактеру који је на реду у другој речи (на почетку, на реду је први карактер са индексом 0). Ако су одговарајући карактери једнаки, прелази се на наредни карактер у обе речи, а иначе само у првој. Циклус треба прекинути када прођемо кроз све карактере бар једне речи. Ако смо прошли кроз све карактере друге речи, закључујемо да се друга реч може добити прецртавањем слова прве, иначе не.

Ниске се могу представити или типом `string` (што је свакако најбоље решење) или у језику C++ низом карактера терминисаним нулом тј. карактером чији је ASCII код нула, а који се често записује као `'\0'` (такво решење је више у духу језика C).

Други поглед на исти овај поступак је да се за сваки карактер друге речи провери да ли постоји у другој речи, при чему се претрага за првим карактером врши од почетка прве речи, а за сваким наредним од позиције иза оне на којој је претходни карактер нађен. Ако се неки карактер не може пронаћи, тада другу реч није могуће добити од прве. Ако се сви карактери друге речи успешно пронађу, онда је другу реч могуће добити од прве. Претрагу карактера можемо вршити или алгоритмом линеарне претраге или библиотечким функцијама типа `string`. У

језику C++ може се употребити метода `find` која враћа специјалну вредност `string::npos` ако карактер није нађен.

Rešenje 02

```
#include <iostream>
using namespace std;

int main() {
  // ucitavamo dve reci u nizove karaktera (cuvajuci jedno dodatno
  // mesto za terminalnu nulu)
  char s1[100 + 1], s2[100 + 1];
  cin >> s1 >> s2;

  // redom prolazimo kroz slova obe reci dok ne dodjemo do kraja jedne
  // od njih (kraj reci je obelezen terminalnom nulom)
  int i = 0, j = 0;
  while (s1[i] != '\0' && s2[j] != '\0') {
    // ako je tekuce slovo prve reci jednako tekucem slovu druge reci
    // onda ga zadrzavamo, a u suprotnom ga precrtavamo
    if (s1[i] == s2[j])
      // ako smo slovo zadrzali, prelazimo na naredno slovo druge reci
      j++;
    // prelazimo na naredno slovo prve reci
    i++;
  }

  // rec se moze dobiti ako i samo ako smo stigli do kraja druge reci
  // (sva slova druge reci smo pronasli u prvoj)
  if (s2[j] == '\0')
    cout << "da" << endl;
  else
    cout << "ne" << endl;

  return 0;
}
```

Rešenje 03

```
#include <iostream>
#include <string>
using namespace std;

int main() {
  // ucitavamo dve reci
  string s1, s2;
  cin >> s1 >> s2;

  // tekuca pozicija u drugoj reci
  int j = 0;
  // prolazimo kroz sva slova prve reci
  for (int i = 0; i < s1.size(); i++) {
    // ako je tekuce slovo prve reci jednako tekucem slovu druge reci,
    // onda ga zadrzavamo, u suprotnom ga precrtavamo
```

```
    if (s1[i] == s2[j])
      j++;
    // ako smo stigli da kraja druge reci, druga rec se moze dobiti od
    // prve tako sto se sva slova do kraja precrtaju
    if (j == s2.size())
      break;
  }

  // rec se moze dobiti samo ako smo stigli do kraja druge reci
  if (j == s2.size())
    cout << "da" << endl;
  else
    cout << "ne" << endl;

  return 0;
}
```

Rešenje 04

```
#include <iostream>
#include <string>
using namespace std;

int main() {
  // ucitavamo dve reci
  string s1, s2;
  cin >> s1 >> s2;

  // tekuca pozicija u prvoj reci
  int i = 0, j;
  // trazimo jedno po jedno slovo druge reci u prvoj
  for (j = 0; j < s2.size(); j++) {
    // pretragu pokrecemo od pozicije i
    int k;
    for (k = i; k < s1.size(); k++)
      // ako smo nasli slovo, prekidamo pretragu
      if (s2[j] == s1[k])
          break;
    // ako smo dosli do kraja prve reci, nismo nasli slovo i prekidamo
    // pretragu
    if (k == s1.size())
      break;
    // naredna pretraga treba da pocne od pozicije k+1
    i = k+1;
  }

  // rec se moze dobiti samo ako smo stigli do kraja druge reci
  if (j == s2.size())
    cout << "da" << endl;
  else
    cout << "ne" << endl;
  return 0;
}
```

Rešenje 05

```
#include <iostream>
#include <string>
using namespace std;

int main() {
  // ucitavamo dve reci
  string s1, s2;
  cin >> s1 >> s2;

  // trazimo jedno po jedno slovo druge rece u prvoj, krenuvsi od pozicije i
  int i, j;
  for (i = 0, j = 0; j < s2.size(); i++, j++) {
    // trazimo tekuce slovo drugoj reci u prvoj, krenuvsi od pozicije i
    i = s1.find(s2[j], i);
    // ako ga nismo nasli, tada prekidamo pretragu
    if (i == string::npos)
      break;
    // prelazimo na sledecu poziciju u obe reci
  }

  // rec se moze dobiti samo ako smo stigli do kraja druge reci
  // (tada je svako njeno slovo pronadjeno u prvoj)
  if (j == s2.size())
    cout << "da" << endl;
  else
    cout << "ne" << endl;
  return 0;
}
```

3. Unose se dva niza karaktera niz1 i niz2. Napisati program kojim ce se iz stringa niz1 izbaciti sva slova koja pojavljuju u nizu niz2. Ispisati niz1.

ULAZ	IZLAZ
aa123bb abcd	123
123abcd aaaa	123bcd

4. 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.

ULAZ	IZLAZ
vatra trava	Jesu anagrami
vatra tavav	Nisu anagrami

Rešenje 01

#include <iostream>

```
#include <algorithm>
using namespace std;

int main()
{
   string s1, s2;
  getline(cin,s1);
  getline(cin,s2);

  sort (s1.begin(), s1.end());
  sort (s2.begin(), s2.end());

  if (s1==s2)
                cout <<"Jeste anagram\n";
        else
                cout <<"Nije anagram\n";


  return 0;
 }
```

Rešenje 02

```
#include <stdio.h>
#include <string.h>
#define MAX 129

void sortirajrec( char s[], int n)
{
        int i,j,min;
        for(i=0;i<n-1;i++)
        {
                min=i;
                for(j=i+1;j<n;j++)
                {
                        if(s[j]<s[min])
                                min = j;
                }
                if(min!=i)
                {
                        char temp;
                        temp = s[min];
                        s[min] = s[i];
                        s[i] = temp;
                }
   }
}
void anagram(char s1[],char s2[] ,int n1,int n2)
{
        if(n1!=n2)
        {
                printf("Nije anagram - nema isti broj slova\n");
```

```
                return;
        }
        int x = strcmp(s1,s2);
        if(!x)
                printf("Jeste anagram\n");
        else
                printf("Nije anagram\n");
}
int main()
{
        char s1[MAX];
        char s2[MAX];
        int n1,n2;

        printf("Unesite dve niske:\n");
        scanf("%s",s1);
        scanf("%s",s2);

        n1= strlen(s1);
        n2= strlen(s2);

        sortirajrec(s1,n1);
        sortirajrec(s2,n2);
        anagram(s1,s2,n1,n2);

}
```

5. Napisati C program koji unosi ceo broj n i ispisuje na standardni izlaz svaku n-tu reč iz datoteke standardnog ulaza.

ULAZ

4

ulaz.txt	izlaz.txt
Kada 123 solja za kafu	za jogurt za
Ekspres lonac, jogurt i mleko	
Spisak za kupovinu	

6. Napisati C program koji prikazuje vertikalni histogram dužina reči koji se pojavljuju na standardnom ulazu.

```
ULAZ
Osnovi programiranja su moj najdrazi predmet na prvoj godini studija.
Vredno sam ucila,odnosno ucio, tokom leta i zato ocekujem da cu
poloziti ispit iz ovog interesantnog predmeta
bez potrebe da se oslonim na nepostene radnje kao sto su
prepisivanje iz ranije
pripremljenih puskica ili uznemiravanje kolege
da mi dozvoli da prepisem njegova resenja.
IZLAZ
  13  |    *
  12  |    *
  11  |    *
  10  |    *
   9  |    *
   8  |    *
   7  |    *           *  *  *
   6  |    *  *        *  *  *
   5  |    *  *        *  *  *        *
   4  |    *  *  *     *  *  *        *
   3  |    *  *  *  *  *  *  *        *
   2  |    *  *  *  *  *  *  *        *
   1  | *  *  *  *  *  *  *  *  *     *
      +--------------------------------
        1  2  3  4  5  6  7  8  9 10 >10
```

```
#include <iostream>
#include <cstdio>

int main()
{
   char c;
   int niz[12]={0};
   bool ok=false;
   int brojac=1;
   int max=0;
   while((c=getchar())!=EOF)
   {
      if(('a'<=c && c<='z') || ('A'<=c && c<='Z'))
      {
         if(ok)
         {
           brojac++;
         }
         else
         {
            brojac=1;
            ok=true;
         }
      }
      else
      {
         if(ok)
         {
            if(brojac>10)
            {
               niz[11]++;
```

```
                if(niz[11]>max)
                {
                   max=niz[11];
                }
             }
             else
             {
                niz[brojac]++;
                if(niz[brojac]>max)
                {
                   max=niz[brojac];
                }
             }
             ok=false;
          }
       }
    }
    for(int i=max;i>0;i--)
    {
       printf("%2d  |",i);
       for(int j=1;j<12;j++)
       {
          printf(" ");
          if(niz[j]>=i)
          {
             printf("*");
          }
          else
          {
             printf(" ");
          }
          printf(" ");
       }
       printf("\n");
    }
    printf("    +----------------------------------\n");
    printf("      1  2  3  4  5  6  7  8  9  10 >10");
    return 0;
}
```

Potprogrami, rekurzija, pokazivači

1. Korišćenjem identiteta $\sqrt{4 \cdot x} = 2 \cdot \sqrt{x}$ napisati rekurzivnu funkciju koja izračunava ceo deo korena datog broja. Napisati C program koji pozivom ove funkcije računa ceo deo korena za brojeve od 1 do 101.

2. Napisati funkciju **void** broj_i_zbir_cifara(**int** a, **int** *brojCifara, **int** *zbirCifara) koja za dati broj a računa od koliko cifara se sastoji broj a i zbir tih cifara. Napisati program koji unosi dva cela broja M i N i ispisuje sve cele brojeve između M i N kojima je broj cifara i zbir cifara iste parnosti (oba neparna ili oba parna.)
ULAZ IZLAZ
90 100 91 93 95 97 99 100

3. Napisati funkciju BrNul(n) koja broji nule u binarnom zapisu prirodnog broja n. Zatim napisati i funkciju Suma, koja za ulazne parametre n i m (prirodni brojevi, m>n) izračunava i vraća vrednost sume:

$$\text{Suma} = \frac{\text{BrNul}(n)}{n} + \frac{\text{BrNul}(n+1)}{n+1} + \frac{\text{BrNul}(n+2)}{n+2} + \ldots + \frac{\text{BrNul}(m)}{m}$$

4. Napisati rekurzivnu funkciju koja izračunava proizvod neparnih cifara neoznačenog celog broja n. Napisati C program koji testira rad ove funkcije.

5. Šta je rezultat rada sledećeg programa? Objasniti!!!

```
#include <stdio.h>
int main()
{
int a[]={0,1,2,3,4};
int* p[]={a,a+1,a+2,a+3,a+4};
int** pp = p;
*pp++;
printf("%d %d %d\n", pp-p, *pp - a, **pp);
pp=p;
++**p;
printf("%d %d %d\n", pp-p, *pp - a, **pp);
signed char c = -128<<2;
printf("%d\n", c );
printf("%d\n", -128<<2 );
signed char c1 = -128>>2;
printf("%d\n", c1 );
return 0;
}
```

6. Napisati rekurzivnu funkciju kojom se proverava da li je broj n prost broj. Napisati C program koji testira rad ove funkcije.

ULAZ	IZLAZ
2	da
4	ne
773	da
923	ne

7. Napisati iterativnu funkciju koja je po funkcionalnosti ekvivalentna sledećoj rekurzivnoj funkciji f.

```
int f(int *a, int n)
{   if (n<=0) return 1;
    else
    {
        if (*a != *(a+n)) return 0;
        else return f(a+1, n-2);
    }
}
```

8. Napisati program kojom vrši sažimanje nizu a prirodnih brojeva dužine n tako što se izbacuju svi prosti brojevi.

9. Ako je dat niz prirodnih prirodnih brojeva, napisati program kojim se vraca najveca duzina uzastopnih prostih brojeva.

ulaz	izlaz
1 2 3 4 5	2
1 2 3 5 7 8 9 11 13 17 18	4

10. Napisati program kojim se u datom nizu pozitivnih brojeva a dužine n zamenjuje vrednost svakog elementa vrednošću prvog sledećeg elementa koji je veći od njega, ako takvog nema, zameni element nulom.
a: 1 9 8 5 9 3 2 4 5
a: 9 0 9 9 0 4 4 5 0

Pretraga i sortiranje

1. *Unikat* je član niza koji se pojavljuje tačno jednom u nizu. Napisati program koji učitava dimenziju niza, a potom i niz celih brojeva i nalazi i ispisuje najveći unikat tog niza. Dimenzija niza je n, $3<n<100$. Voditi računa o vremenskoj složenosti programa.

PRIMER 1	PRIMER 2
ULAZ	ULAZ
5	16
1 -2 36 9 1	1 2 3 2 1 -8 3 67 1 13 28 1 67 13 4 6
IZLAZ	IZLAZ
36	28

2. Poznati prodavac jabuka Steva izabrao je **n** jabuka koje će odneti na pijacu na prodaju. Kada je stigao na pijacu, uvideo je da toga jutra je konkurencija dosta jaka. Mudri Steva je odlučio da svoju ponudu pospeši tako što će jabuke na svojoj tezgi aranžirati na specijalan način: najlakša jabuka će stajati krajnje levo, sledeća najlakša jabuka će stajati u krajnje desno na tezgi. Ovaj proces se nastavlja sve dok Steva ne postavi sve jabuke, ali tako da najteža jabuka bude u sredini.
Napišite programa, koji prikazuje raspored Stevinih jabuka.
U prvoj liniji standardnog ulaza će biti predstavljen broj lubenica n ($3<n<50$). Nakon toga sledi n pozitivnih celih brojeva (ne većih od 100), u kojima je izražena masa svake jabuke u kilogramima. Na standardni izlaz ispisati u jednoj liniji masu jabuka posle aranžiranja.

PRIMER

ULAZ	IZLAZ
5	1 3 5 4 2
1 2 3 4 5	

3. U restoranu je poznato vreme dolaska i vreme odlaska svakog gosta u toku jednog dana (ss:mm) od 00:00 do 23:59. U prvoj liniji standardnog ulaza se nalazi broj gostiju N, a zatim za svakog gosta vreme dolaska ss:mm i vreme odlaska ss:mm, odvojeni jednim blanko znakom. Redosled gostiju u fajlu je proizvoljan. Prikazati na standardnom izlazu sve periode u toku dana kada u restoranu nije bilo gostiju.

4. U ravni je dato N materijalnih tačaka, za svaku tačku poznata nam je njena koordinata (x_i,y_i) i masa m_i. Možete koristiti 3 niza x,y,m. Od nekog momenta tačka najmanje mase iščezava predajući svoju masu njoj

najbližoj tački. Proces se nastavlja dok ne ostane jedna tačka. Kreirati program kojim se realizuje ovaj proces i određuje tačka kojoj sve ostale predaju masu.

5. U datom nizu brojeva treba pronaći dva broja koja su na najmanjem rastojanju. Niz se zadaje sa standardnog ulaza, sve do kraja ulaza, ali neće sadržati više od 256 i manje od 2 elemenata. Na izlaz ispisati razliku pronađena dva broja. Uputstvo: Prvo sortirati niz.

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

Test 2 Ulaz: 21 654 65 123 65 12 61
Izlaz: 0

Test 3 Ulaz: 34 30
Izlaz: 4

6. 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 1000 i kraći od jednog 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

Test 3 Ulaz: 123
Izlaz: 123

7. Napisati funkciju int merge(int *niz1, int dim1, int *niz2, int dim2, int *niz3, int dim3) koja prima dva sortirana niza, i na osnovu njih pravi novi sortirani niz koji sadrži elemente oba niza. Treća dimenzija predstavlja veličinu niza u koji se smešta rezultat. Ako je ona manja od potrebne dužine, funkcija vraća -1 kao indikator neuspeha, inače vraća 0. Napisati zatim program koji testira ovu funkciju. Nizovi se unose sa standardnog ulaza sve dok se ne unese 0. Može se pretpostaviti da će njihove dimenzije biti manje od 256.

Primer 1 Interakcija sa programom:
Unesite elemente prvog niza: 3 6 7 11 14 35 0
Unesite elemente drugog niza: 3 5 8 0
IZLAZ
3 3 5 6 7 8 11 14 35

Primer 2 Interakcija sa programom:
Unesite elemente prvog niza: 1 4 7 0
Unesite elemente drugog niza: 9 11 23 54 75 0
IZLAZ
1 4 7 9 11 23 54 75

8. U datoteci pesme.txt nalaze se informacije o gledanosti pesama na Youtube-u. Svaki red datoteke sadrži informacije o gledanosti pesama u formatu izvođač - naslov, broj gledanja.

Napisati program koji učitava informacije o pesmama i vrši sortiranje pesama u zavisnosti od argumenata komandne linije na sledeći način:
- nema opcija, sortiranje se vrši po broju gledanja;
- prisutna je opcija -i, sortiranje se vrši po imenima izvođača;
- prisutna je opcija -n, sortiranje se vrši po naslovu pesama.

Na standardni izlaz ispisati informacije o pesmama sortiranim na opisani način.

Test 1 Pokretanje: racunaj

pesme.txt
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

Test 2 Pokretanje: racunaj -i

pesme.txt
5
Ana - Nebo, 2342
Laza - Oblaci, 29
Pera - Ptice, 327
Jelena - Sunce, 92321
Mika - Kisa, 5341

Izlaz:
Ana - Nebo, 2342
Jelena - Sunce, 92321
Laza - Oblaci, 29
Mika - Kisa, 5341
Pera - Ptice, 327