

**CS103 - DOMAĆI ZADATAK 02.**

Prilikom slanja domaćeg zadatka svom asistentu neophodno je da ispunite sledeće:

* Subject mail-a mora biti CS103-DZbr. Za ovaj domaći zadatak - CS103-DZ02
* Sve fajlove, koji su deo rešenja zadataka, arhivirati (zip, rar, …)
* Poželjno je uraditi i printscreen koda pre pokretanja programa
* U prilogu maila treba da se nalazi arhiva projekta koji se ocenjuje, imenovana na sledeći način: CS103-DZbr-BrojIndeksa-ImePrezime*. Na primer, CS103-DZ02-9999-VeljkoGrkovic.zip*
* Telo mail-a treba da ima pozdravnu poruku

**Molimo sve studente da se pridržavaju navedenog, inače zadaci neće biti pregledani i ocenjeni.**

Studenti iz Beograda i online studenti zadatke na ocenjivanje šalju mail-om na adresu [lazar.mrkela@metropolitan.ac.rs](mailto:lazar.mrkela@metropolitan.ac.rs)

Studenti iz Niša zadatke na ocenjivanje šalju mail-om na adresu  
[nikola.dimitrijevic@metropolitan.ac.rs](mailto:veljko.grkovic@metropolitan.ac.rs)

**Rok za slanje domaćih zadataka:**

* **Tradicionalni studenti: sedam dana od dana održavanja vežbe na koju se odnosi domaći zadatak. Nakon toga, broj poena se umanjuje za 50%. Krajnji rok za slanje rešenja je deset dana pre termina ispita u kome želite da polažete ispit.**
* **Studenti na online nastavi, kao i studenti kojima je predmet prenet u narednu godinu ili je diferencijalni: najkasnije deset dana pre termina ispita u kome želite da polažete ispit.**

*Svaki student radi po dva zadatka sa spiska.* ***Prvi zadatak se radi u programskom jeziku C/C++.*** ***Rešenje drugog zadatka poslati kao .doc ili .txt fajl.*** Ostali zadaci mogu da posluže za vežbanje i pripremu ispita, ali ih ne šaljete na pregled asistentima.

Redni broj prvog zadatak dobijate tako što vaš broj indeksa podelite sa 15, a dobijeni ostatak pri deljenju uvećate za 1. Redni broj drugog zadataka dobijate kada na dobijeni rezultat dodate 15.   
Na primer:   
Broj indeksa 2378  
2378 % 15 + 1 = 9 – student radi 9. zadatak.   
Dobijeni rezultat -> **9** + 15 = 24 – student radi 24. zadatak iz druge grupe.

1. [C/C++] Dat je niz prirodnih brojeva dužine ***n***. Za dati prirodni broj ispitati da li postoje dva elementa niza, čija je suma jednaka datom broju ***s***. (Traži se algoritam je složenosti O(nlogn)).
2. [C/C++] Data je zgrada sa n spratova. Vama su na raspolaganju dva tanjira, a potrebno je odrediti prvi sprat sa koga se lome tanjiri u što manjem broju bacanja. Vi možete bacati tanjire sa nekih spratova. Ukoliko se oni polome više ih ne možete upotrebljavati. (Traženo rešenje mora biti najmanje složenosti)
3. [C/C++] Naći broj brojeva iz segmenta [A,B] čiji je zbir cifara paran. (Traženo rešenje mora biti najmanje složenosti)

*Ulaz: Izlaz:  
5 15 5*

1. [C/C++] Ispitati da li je dati prirodan broj *n prost*. Ukoliko nije, faktorisati ga (u obliku p1x p2x …pkx). (Traženo rešenje mora biti najmanje složenosti)

*Ulaz Izlaz  
 n = 4256 NE  
 4265 = 225371*

1. [C/C++] Za date prirodne brojeve *a, n, m* izračunati an mod m (traži se rešenje logaritamske složenosti).

*Ulaz Izlaz  
 a = 7 n = 8 m = 1000 801 (tj. 5764801 mod 1000)*

1. [C/C++] Dat je broj *n*. Odrediti da li se on može podeliti na dva dela (povlačenjem crte između dve cifre), tako da oba dobijena broja budu prosta. (Traženo rešenje mora biti najmanje složenosti)

*Ulaz Izlaz  
 223 2|23  
 123 NE*

1. [C/C++] Data je kvadratna matrica popunjena nulama i jedinicama. Odrediti najveći kvadrat sastavljen samo od 0. (Traženo rešenje mora biti najmanje složenosti)

*Ulaz Izlaz  
 5 3  
 0 0 1 1 1   
 0 0 0 0 0  
 1* ***0 0 0*** *1  
 1* ***0 0 0*** *0  
 1* ***0 0 0*** *1*

1. [C/C++] Odrediti minimalan broj sastavljen samo od neparnih cifara, koji po *modulu n daje ostatak m*. (Traženo rešenje mora biti najmanje složenosti)

*Ulaz Izlaz  
 17 57  
 6*

1. [C/C++] Dat je niz celih brojeva, dužine *n*. Odrediti podniz datog niza čiji je zbir elemenata maksimalan, a u kome nema susednih elemenata. (Traženo rešenje mora biti najmanje složenosti)

*Ulaz Izlaz  
 8 20  
 4 -1 3 8 7 -1 -5 6 (****4*** *-1* ***3*** *8* ***7*** *-1 -5* ***6*** *– selektovani brojevi su dato rešenje).*

1. [C/C++] Dat je niz *A* prirodnih brojeva dužine *n*. Treba niz *A* podeliti u dve grupe, tako da je razlika zbirova elemenata u grupama minimalna. Štampati jednu takvu grupu. (Traženo rešenje mora biti najmanje složenosti)

*Ulaz Izlaz  
 n = 4 2 4 2 | 7  
 A = 2 4 2 7*

1. [C/C++] Ispred blagajne je poređano *n* ljudi sa svojim visinama h1, h2,…, hn, redom. Odrediti broj ljudi koje vidi blagajnik. Blagajnik vidi neku **osobu x** ukoliko su sve osobe u redu ispred x niže od osobe y. (Traženo rešenje mora biti najmanje složenosti)

*Ulaz Izlaz  
 n = 8 4  
 h = 5 5 8 7 11 12 5 11 (****5*** *5* ***8*** *7* ***11******12*** *5 11 – osobe koje vidi blagajnik)*

1. [C/C++] U fajlu je dat niz od n brojeva. Sa dva prolaska kroz niz odrediti da li postoji broj koji se pojavljuje više od n/2 puta. (Traži se algoritam je složenosti O(n)).

*Ulaz Izlaz  
 n = 9 broj 5  
 5 7 1 1 5 5 2 5 5*

1. [C/C++] Naći sve parove prostih brojeva (a,b) za koje je a < b i za koje važi: a + b je takođe prost broj čija vrednost ne prelazi n. Na ulazu je dat prirodan broj n. U prvoj liniji izlaza ispisati ukupan broj traženih parova. Zatim ispisati parove. (Traženo rešenje mora biti najmanje složenosti)

*Ulaz Izlaz  
 5 1  
 2, 3*

1. [C/C++] Data je matrica dimenzija *nxm*, za koju važi da su sve vrste sortirane sa leva udesno i sve kolone sortirane odozgo na dole. Odrediti da li u matrici postoji *broj x* u vremenu O(n + m) I ispisati njegove koordinate. (Traženo rešenje mora biti najmanje složenosti)

*Ulaz Izlaz  
 n = 3, m = 4, x = 8, 3, 2  
 matrica:  
 1 4 6 7  
 2 5 9 12  
 3 8 10 13*

1. [C/C++] Na žurci je došlo *n* momaka i *m* devojaka. Momak i devojka mogu plesati samo ukoliko je ime devojka anagram momkovog imena. Koliki je maksimalni broj parava koji mogu igrati u jednom trenutku? (Traženo rešenje mora biti najmanje složenosti)

*Ulaz Izlaz  
 n=3 m =2 1  
 marko ivaan nikola  
 ivana marija*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Druga grupa zadataka: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Oceniti vremensku složenost sledećeg algoritma:

*for (int i = 1; i <= 15; i++) {*

*for (int j = 1; j <= n; j++) {*

*i/2;*

*n=n/2*

*}*

*}*

1. Oceniti vremensku složenost sledećeg algoritma:

*bool ContainsValue(String[] strings, String value){*

*for(int i = 0; i < strings.Length; i++){*

*if(strings[i] == value){*

*return true;*

*}*

*}*

*return false;*

*}*

18. Oceniti vremensku složenost sledećeg algoritma:

*bool ContainsDuplicates(String[] strings){*

*for(int i = 0; i < strings.Length; i++){*

*for(int j = 0; j < strings.Length; j++){*

*if(i == j){*

*continue;*

*}*

*if(strings[i] == strings[j]){*

*return true;*

*}*

*}*

*}*

*return false;*

*}*

19. Oceniti vremensku složenost sledećeg algoritma:

*int CompareSmallestNumber (int array[]){*

*int x, curMin;*

*curMin = array[0];*

*for (x = 1; x < 10; x++){*

*if( array[x] < curMin) {*

*curMin = array[x];*

*}*

*}*

*return curMin;*

*}*

20. Oceniti vremensku složenost sledećeg algoritma:

*for (i = 0; i < N; i++) {*

*for (j = 0; j < N; j++) {*

*t++;*

*}*

*}*

*for (k = 0; k < N; k++) {*

*t++;*

*}*

21. Oceniti vremensku složenost sledećeg algoritma:

*for (i = 0; i < N; i++) {*

*for (j = N; j > i; j--) {*

*k=2\*k;*

*}*

*}*

22. Oceniti vremensku složenost sledećeg algoritma:

*void modifyArray(int a[], int n){*

*int max = a[0];*

*for (int i = 1; i < n/2; i++){*

*if (max < a[i])*

*max = a[i];*

*}*

*for (int j = 1; j <= n \* n; j++){*

*++max;*

*}*

*}*

23. Oceniti vremensku složenost sledećeg algoritma:

*for (int i = 1; i < n; i++){*

*n=n/3;*

*}*

*for (int j = 1; j <= n ; j++){*

*k=k+j;*

*}*

24. Oceniti vremensku složenost sledećeg algoritma:

*int sum;*

*float avarage;*

*for (int i = 0; i < n ; i ++) {*

*a [ i ] = i ;*

*}*

*for (int i = 0; i < n ; i ++) {*

*sum+=a[i]*

*}*

*avarage=sum/n;*

25. Oceniti vremensku složenost sledećeg algoritma:

*for (int i = 0; i < n ; i ++) {*

*for(int j=0; j<m; j++)*

*m=m/2;*

}

26. Oceniti vremensku složenost sledećeg algoritma:

*for (int i = 0; i < n ; i ++) {*

*for(int j=0; j<n; j++){*

*for(int k=1; k<n; k=k\*2){*

*k++; j++;*

*}*

*}*

*}*

27. Oceniti vremensku složenost sledećeg algoritma:

*for ( int i = 0; i < n; i++ ) {*

*for ( int j = 0; j < n; j++ ) {*

*c[i][j] = 0;*

*for ( int k = 0; k < n; k++ ) {*

*c[i][j] += a[i][k] \* b[k][j];*

*}*

*}*

*}*

28. Oceniti vremensku složenost sledećeg algoritma:

*for ( int i = 0; i < n; i++ ) {*

*for ( int j = 0; j < n; j++ ) {*

*c[i][j] = 0;*

*}*

*}*

29. Oceniti vremensku složenost sledećeg algoritma:

*int maxSubsequenceSum(int a[], int n){*

*int thisSum = 0, maxSum = 0, i = 0;*

*for( int j = 0; j < n; j++){*

*thisSum += a[j];*

*if( thisSum > maxSum){*

*maxSum = thisSum;*

*seqStart = i;*

*seqEnd = j;*

*}else if( thisSum < 0) {*

*i = j + 1;*

*thisSum = 0;*

*}*

*}*

*return maxSum;*

*}*

30. Oceniti vremensku složenost sledećeg algoritma:

*for ( int i = 0; i < n; i++ ) {*

*for ( int j = 0; j < n; j++ ) {*

*if (a[ j ] > a[ j + 1 ]) {*

*int t;*

*t = a[j];*

*a[j] = a[ j + 1 ];*

*a[ j + 1 ] = t;*

*}*

*}*

*}*