CS203 - 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 zadatke na ocenjivanje Šalju mail-om na adresu
lazar.mrkela@metropolitan.ac.rs
Studenti iz Niša zadatke na ocenjivanje Šalju mail-om na adrese
lazarevic.uros@metropolitan.ac.rs
jovana.jovanovic@metropolitan.ac.rs
Online studenti zadatke na ocenjivanje Šalju mail'om na adresu
lazarevic.uros@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 jedan zadatak sa spiska. ***Rešenje 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 = – student radi 9. zadatak.

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
      }
}
```

2. 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;
}
```

3. 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;
}
```

4. 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;
}
```

5. 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++;
      }
```

6. Oceniti vremensku složenost sledećeg algoritma:

```
for (i = 0; i < N; i++) { for (j = N; j > i; j--) { k=2*k;
            }
      }
```

7. 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;
            }
      }
```

8. 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;
      }
```

9. 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;
```

10. Oceniti vremensku složenost sledećeg algoritma:

```
      for (int i = 0; i < n ; i ++) {
            for(int j=0; j<m; j++) m=m/2;
      }
```

11. 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++;
```

```
            }
      }
}
```

12. 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];
            }
      }
}
```

13. Oceniti vremensku složenost sledećeg algoritma:

```
for ( int i = 0; i < n; i++ ) { for
      ( int j = 0; j < n; j++ ) {
      c[i][j] = 0;
      }
}
```

14. 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;
}
```

15. 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;
            }
        }
    }
```