LJIR 2012.

6. zadatak (8 bodova)

a) (4 boda) Odredite apriornu složenost programskog odsječka i detaljno obrazložite odgovor. O kojem algoritmu je riječ?

```
void StoJaRadim (int A [], int N) {
  int i, j;
  int pom;
  for (i = 1; i < N; i++) {
    pom = A[i];
    for (j = i; j >= 1 && A[j-1] > pom; j--)
      A[j] = A[j-1];
    A[j] = pom;
  }
}
```

b) (4 boda) Odredite apriornu složenost programskih odsječaka i detaljno obrazložite odgovore:

```
1) int nekaFunkcija(int n) {
     int br = 0;
     int nasao = 0;
     do {
       br++;
       if (br == korak)
         nasao = 1;
       n /= 2;
     } while (n > 0);
     return br;
   }
```

```
2)  nasao = 0;
   for (double i=0; i<n; i++)
     for (double j=0; j<n-i; j++)
       if (i==j && a[i][j]==trazeni)
         nasao = 1;
```

Zadatak 6. (8 bodova)

a) (4 boda)
$O(n^2)$
Ovo je sortiranje umetanjem („Insertion sort"). Postoje dva dijela ulaznog statičkog polja A: sortirani i nesortirani. U svakom koraku algoritma sortirani dio se proširuje tako da se u njega na ispravno mjesto ubaci prvi element iz nesortiranog dijela polja. Vanjska petlja služi za određivanje granice sortiranog dijela, dok unutarnja ubacuje element u sortirani niz i pomiče ostale elemente. Složenost i u najgorem slučaju (kad je niz sortiran naopako) je $O(n^2)$.

b) (2 boda)
$O(\log_2 n)$
Koristi se do-while petlja i provjeravanje uvjeta (if naredba) nema utjecaja na duljinu izvođenja algoritma. Algoritam uzastopce cjelobrojno dijeli ulazni broj s 2 dok god je rezultat veći od 0. Dakle, petlja će se izvršiti onoliko puta koliko je n potencija broja 2, odnosno točnije, $\log_2 n + 1$ puta. Naredbe prije i poslije do-while petlje imaju konstantu složenost.

c) (2 boda)
$O(n^2)$
Kroz petlje se prolazi uvijek, bez obzira je li uvjet unutar petlje zadovoljen ili ne, jer nema break naredbe. Broj izvršavanja 1 (n=1), 55 (n=10), 5050 (n=100), 500500 (n=1000), itd. Točniji opis složenosti je $O(½\ n^2)$.

JIR 2013.

Zadatak 4. (10 bodova)

Odredite asimptotsku složenost sljedećih algoritama:

a)
```
s=0;
for(i=1;i<=n;i*=2)
      for(j=1;j<=n;j++)
            s+=radi(n,j)+radi(n,i);
```

b)
```
s=0;
for(i=1;i<=n;i*=2)
      for(j=1;j<=n;j++)
            s+=radi(j,n)+radi(j,i);
```

uz pretpostavku da je asimptotska složenost funkcije radi(x,y) jednaka Θ(radi(x,y))=x. Obrazložite odgovor.

Zad4.
a) $2n^2\log_2 n$ b) $n(n+1) \log_2 n$

JIR 2014.

Zadatak 3. (8 bodova)

Odredite gornju granicu (veliki O) složenosti i asimptotsku složenost zadanih odsječaka uz pretpostavku konstantne asimptotske složenosti funkcije f(x) i da je n >> 1. Obrazložite odgovor.

a)
```
a=0;
for(i=0;i<n;i++)
      for(j=n;j>i;j--)
            a+=f(j);
```

O:	asimptotska:

b)
```
a=0;
for(i=n; i>1; i=(int)sqrt(i))
      for(j=1;j<i;j*=2)
            a+=f(j);
```

O:	asimptotska:

Zadatak 3. (8)

a) broj iteracija je n + (n – 1) + (n – 2) + … + 1, dakle:
$O(n^2)$;
$\Theta((n+1)*n/2)$

b) unutarnja petlja je logaritamske složenosti prema varijabli i, a i poprima po vrijednosti n, $\sqrt{n}$, $\sqrt{\sqrt{n}}$,… dakle:
$\Theta(\log_2 n^{1/2} + \log_2 n^{1/4} + \log_2 n^{1/8} + \ldots) = \Theta((1 + ½ + ¼ + \ldots) * \log_2 n)) \sim= \Theta(2 \log_2 n)$
tj. $O(\log_2 n)$;

DIR 2014.

Zadatak 3. (12 bodova)

a) Koja je apriorna složenost zbrajanja dvaju n-znamenkastih brojeva na papiru?

b) Poredajte uzlazno sljedeće složenosti (nakon što su poredani, među izrazima mora biti znak **>**,**<** ili **=**):
$O(2^n)$, $O(n^{1.5})$, $O(n)$, $O(n \log_2 n)$, $O(n \ln n)$, $O(\sqrt{n})$, $O(n!)$, $O(n^n)$, $O(n^{1000})$, $O(3^n)$

c) Koja je apriorna složenost posljednjih triju linija sljedećeg programskog odsječka:

```
double n;
/* ... odsječak u kojem se n postavlja
na neku vrijednost ...*/
while (n>1) {
      n*=0.999;
}
```

Zadatak 3. (12)

a) $O(n)$

b) $O(\sqrt{n}) < O(n) < O(n \log_2 n) = O(n \ln n) < O(n^{1..5}) < O(n^{1000}) < O(2^n) < O(3^n) < O(n!) < O(n^n)$

c) $O(\log n)$

ZI 2015.

Zadatak 2. (6 bodova)

Koja je složenost sljedećeg odsječka u O i Ω notaciji u ovisnosti o argumentu n?

```
void funct(int n) {
  int i,j;
  for (i=n;i>0;i/=2) {
    for (j=0;j<i;j++) {
      printf("%d\n",j%2);
    }
  }
}
```

O notacija (2 boda):

Ω notacija (1 bod):

```
void funct(int n) {

  int i,j;
  if (n<20)
    fact(n);
  else {
    if (n%2) {
      for (i=0;i<n;++i) {
        for (j=0;j<i;++j) {
          printf("%d\n",j%2);
        }
      }
    }
    else {
      for (i=0;i<n;++i) {
        printf("%d\n",i%2);
      }
    }
  }
}
```

O notacija (2 boda):

Ω notacija (1 bod):

(napomena: funkcija fact računa faktorijel argumenta, a modulo operator u argumentu printf-a osigurava jednako vrijeme izvođenja printf-a bez obzira na veličinu argumenta)

Zadatak 2. (6 bodova)

O(n) (**2B**) (jer (N+N/2+N/4+...+1) = 2N, pa je i dolje omega od N)	O(n^2) (**2B**) (poziv fact ne ulazi u računicu za velike n-ove, pa tako ni u O notaciju)
Ω(n) (**1B**)	Ω(n) (**1B**)

JIR 2015.

Zadatak 4. (10 bodova)

Odredite a priori složenost (u O-notaciji) zadanih odsječaka programskog koda uz pretpostavku konstantne asimptotske složenosti funkcije f(n) i da je n >> 1. Obrazložite odgovor.

a) (**4 boda**)

```
a = 0;
for(i=1; i<=n; i++) {
  for(j=1; j<=i; j++) {
    a += f(j);
  }
}
```

b) (**6 bodova**)

```
a = 0;
for(i=1; i<=n; i++) {
  for(j = pow(n, 1/(i*i)); j>=1; j/=2) {
    a += f(i);
  }
}
```

Naputak: $\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6} \approx 1{,}64$

Zadatak 4. (10 bodova)

(Zadatak sličan zadatku od prije 2 godine, izazov za studente je da nisu baš napamet naučili ono rješenje)

a) $O(n^2)$

b) $O(n) + O(\log(n^{1/1}) + \log(n^{1/4}) + \log(n^{1/9}) + \cdots) = O(n) + O\left(\left(1 + \frac{1}{4} + \frac{1}{9} + \cdots\right)\log(n)\right) = O(n) + O(\log(n)) = O(n)$

MI 2016.

Zadatak 2. (6 bodova)

Odredite vrijeme izvođenja u O, Ω i, ako je moguće, Θ notaciji za programski odsječak u a) dijelu zadatka i za funkciju **rekurzija** u b) dijelu zadatka. Ako se vrijeme izvođenja u Θ notaciji ne može odrediti, navedite tako u rješenju. Rješenja upišite u tablice pored zadatka.

a)

```
/* A je polje n cijelih brojeva */
if (n <= 10) {
  g(n); /* obavlja se u Θ(n) vremenu */
}
else {
  for (i = n - 1; i > 0; i--) {
    if (A[i] > A[i - 1]) {
      g(i);
    }
  }
}
```

O	
Ω	
Θ	

b)

```
void rekurzija(int n) {
  int i;
  if (n == 0) {
    return;
  }
  for (i = 0; i < n; i++) {
    rekurzija(i);
  }
}
```

O	
Ω	
Θ	

2. zadatak

a) $O(n^2)$, $\Omega(n)$ (najbolji slučaj: silazno sortirano polje; najgori slučaj: uzlazno sortirano polje)

b) $O(2^n)$, $\Omega(2^n)$, $\Theta(2^n)$

LJIR 2016.

Zadatak 4. (10 bodova)

Odredite vrijeme izvođenja u O, Ω i, ako je moguće, Θ notaciji za programski odsječak u a) dijelu zadatka i za funkciju **det** u b) dijelu zadatka. Ako se vrijeme izvođenja u Θ notaciji ne može odrediti, navedite tako u rješenju. Rješenja upišite u na to predviđeno mjesto ispod zadatka.

a)

```
/* A je polje n*n cijelih brojeva,
   sve varijable su tipa int, funkcija f
   ima asimptotsku složenost Θ(n)  */

tSum=0;
for(i=0;i<n; i++)
  for (j=1; j<=n; j*=2)
    tsum+=A[n*i+j-1]*f(n,i,j);

for(i=0;i<n;i++){
  sSum=A[i*n+0]+A[i*n+1]+A[i*n+2];
  if (sSum>0){
     for(k=0;k<n;k++)
        sSum+= A[n*i+k]*f(n,i,k+1);
     tSum += sSum
  }
}
```

O ______ Ω _______ Θ_______

b)

```
/* A je polje n*n double brojeva, a
   pretpostavljamo da funkcije
   preuredi i free imaju asimptotsku
   složenosti Θ(1).  */

double det(double *A, int n){
   double sm,s, *pom;
   int j;
   if (n == 1) return A[0];
   s = 1; sm = 0;
   for (j = 0; j < n; j++){
      pom = preuredi(A, n, j+1);
      sm += (s*A[j] * det(pom, n - 1));
      s = -s; free(pom);
   }
   return sm;
}
```

O ______ Ω _______ Θ_______

4. Zadatak

a) U najgorem slučaju je O(n3), u najboljem Ω(n2 log(n)), a asimptotska ne postoji

b) Sve tri složenosti su n!

neka je T(n)=složenost za det(n)

T(n)=2*n (množenja)+ n (zbrajanja) + n*T(n-1)= Θ(n) + n*T(n-1) =

= Θ(n)+n (Θ(n-1) + (n-1)*T(n-2)) = Θ(n) + Θ(n*(n-1))+n*(n-1)*T(n-2)

= Θ(n*(n-1))+n*(n-1)*(Θ(n-2)+ (n-2)*T(n-3)) = (sličan račun)

= Θ(n*(n-1)*(n-2)) + n(n-1)(n-2)T(n-3) = analognim računom se dođe do faktorijela

DIR 2016.

Zadatak 5. (6 bodova)

Zadano je polje od **n** članova tipa int. Odredite vrijeme izvođenja u Θ notaciji potrebno za sortiranje polja sljedećim algoritmima posebno razmatrajući **prosječan** (očekivani) te **najgori** slučaj za svaki od algoritama:

	Prosječan (očekivani) slučaj	Najgori slučaj
Sortiranje umetanjem (*insertion sort*)	Θ______	Θ______
quicksort	Θ______	Θ______

5. zadatak (6 bodova)

	Prosječan (očekivani) slučaj	Najgori slučaj
Sortiranje umetanjem (*insertion sort*)	$\Theta(n^2)$	$\Theta(n^2)$
Quicksort	Θ(n log n)	$\Theta(n^2)$

MI 2017.

Zadatak 2. (7 bodova)

Odredite vrijeme izvođenja u O, Ω i, gdje je moguće, Θ notaciji za funkcije **f1** i **f2**. Ako se vrijeme izvođenja u Θ notaciji ne može odrediti, navedite tako u rješenju. Rješenja upišite u tablice pored zadataka.

a)

```
/* A je polje n cijelih brojeva.
 * Funkcije za sortiranje sortiraju niz uzlazno,
 * a koriste algoritam naveden u imenu funkcije.
*/
void f1(int *A, int n) {
  insertionSort(A, n);
  mergeSort(A, n);
}
```

O	
Ω	
Θ	

b)

```
void f2(int n) {
  int i;
  if (n == 1) {
    return;
  }
  for (i = 0; i < n; i++) {
    f2(n - 1);
  }
}
```

O	
Ω	
Θ	

2. zadatak (7 bodova)

a) $O(n^2)$, Ω(n log n) (najbolji slučaj: polje je već uzlazno sortirano; najlošiji slučaj: silazno sort. polje)

b) O(n!), Ω(n!), Θ(n!)

JIR 2017.

Zadatak 4. (12 bodova)

Odredite, gdje je moguće, vrijeme izvođenja O, Ω i Θ notaciji za funkcije **f1** i **f2**. Ako se vrijeme izvođenja ne može odrediti, navedite tako u rješenju. Rješenja upišite u tablice pored zadataka.

a)

```
void f1(int n) {
  int i;
  for(i=n; i>0; i/=2) {
    printf("%d\n", i%6);
  }
  if(i>0)
    f1(i*2);
}
```

O	
Ω	
Θ	

b)

```
void f2(int A[], int n) {
  int i, j, pom;
    for (i = 1; i < n; i++) {
      pom = A[i];
      for (j = i; j >= 1 && A[j-1] > pom; j -= 1) {
         A[j] = A[j - 1];
      }
      A[j] = pom;
    }
}
```

O	
Ω	
Θ	

4. zadatak (12 bodova)

a)
// dok se izađe iz petlje bit će i=0 te se rekurzivni poziv niti jednom neće izvesti
O (log n)
Ω (log n)
Θ (log n)

b)
// ovo je insertion sort
O (n^2)
Ω (n)
Θ nije definirano

DIR 2017.

Zadatak 5. (12 bodova)

Odredite, gdje je moguće, vrijeme izvođenja u O, Ω i Θ notaciji za funkcije **f1** i **f2**. Ako se vrijeme izvođenja ne može odrediti, navedite tako u rješenju. Rješenja upišite u tablice iznad zadataka.

O	
Ω	
Θ	

O	
Ω	
Θ	

a) Izrazite složenost u ovisnosti o **n=b-a+1**
Za funkciju gfun(m) vrijedi
O(gfun(m))=Ω(gfun(m))=Θ(gfun(m))=m

```
int f1(int a, int b){
   int s;
   s=gfun(b-a+1);
   s+=f1(0,n/2)+f1(n/2+1,n)
   return s;
}
```

b) Izrazite složenost u ovisnosti o **n.**

```
int f2(int *A, int n){
  int i,j,k,s;
  for(k=0; k<n; i++){
    if (n % 2==0){
      for(i=0; i<k; i++)
        for(j=0; j<k; j++)
          s+=A[i*k+j]*A[j*k+i];
    }
    else{
      for(i=0; i<2*k; i++)
        s+= A[i*k]*A[(i+1)*k];
    }
  }
  return s;
}
```

a) u sva tri slučaja složenost se ne može odrediti jer se algoritam nikad ne zaustavlja, odnosno nije definiran kriterij zaustavljanja u rekurziji.

b)
u sva tri slučaja složenost se ne može odrediti jer se algoritam nikad ne zaustavlja. Naime, vanjska petlja ima iterator k, a k se ne mijenja nigdje u petlji (obrati pažnju na i++).