ДРУШТВО МАТЕМАТИЧА СРБИЈЕ

ДРЖАВНИ СЕМИНАР ЗА НАСТАВНИКЕ МАТЕМАТИКЕ И РАЧУНАРСТВА 2020.

ТЕМА 9:

АЛГОРИТМИ КОМБИНАТОРИКЕ И ПРОБЛЕМИ ТЕОРИЈЕ БРОЈЕВА У ТАКМИЧАРСКОМ ПРОГРАМИРАЊУ

РЕАЛИЗАТОР ТЕМЕ:
Државна комисија за информатичка такмичења ученика основних школа (Јелена Хаџи-Пурић)

Резиме

Аутоматска обрада велике количине података нужно захтева коришћење алгоритама који су од давнина познати математичарима, али је њихова ефикасност еволуирала током времена.
Показаћемо да решавање такмичарских задатака је важан наставни проблем, јер представља парадигму решавања проблема уопште.
Циљ нам је да укажемо на шири избор метода, али и на предност коришћења одређених програмских језика и структура података како би обогатили имплементациону моћ наставника и ученика.

На такмичењима из рачунарства у Србији се користи 6 програмских језика: Ц, Ц++, Јава, Пајтон, Паскал, Ц#. Зато ћемо дати упоредни приказ имплементације алгоритама комбинаторике и теорије бројева у шест програмских језика и анализирати ефикасност имајући на уму временску сложеност и неопходне меморијске ресурсе.

БЕОГРАД,
08. – 09. 02. 2020.

Да се подсетимо: Временска сложеност алгоритма

Алгоритми чија је сложеност одозго ограничена полиномијалним функцијама сматрају се ефикасним.

$n/f(n)$	$\log n$	$\sqrt{n}$	n	$n \log n$	n^2	n^3
10	0,003 μs	0,003 μs	0,01 μs	0,033 μs	0,1 μs	1 μs
100	0,007 μs	0,010 μs	0,1 μs	0,644 μs	10 μs	1 ms
1,000	0,010 μs	0,032 μs	1,0 μs	9,966 μs	1 ms	1 s
10,000	0,013 μs	0,1 μs	10 μs	130 μs	100 ms	16,7 min
100,000	0,017 μs	0,316 μs	100 μs	1,67 μs	10 s	11,57 dan
1,000,000	0,020 μs	1 μs	1 ms	19,93 μs	16,7 min	31,7 god
10,000,000	0,023 μs	3,16 μs	0,01 s	0,23 s	1,16 dan	3×10^5 god
100,000,000	0,027 μs	10 μs	0,1 s	2,66 s	115,7 dan	
1,000,000,000	0,030 μs	31,62 μs	1 s	29,9 s	31,7 god	

Алгоритми чија је сложеност ограничена одоздо експоненцијалном или факторијелском функцијом се сматрају неефикасним.

$n/f(n)$	2^n	$n!$
10	1 μs	3,63 ms
20	1 ms	77,1 god
30	1 s	$8,4 \times 10^{15}$ god
40	18,3 min	
50	13 dan	
100	4×10^{13} god	

1. Ленкин тата ради између 14 часова и поноћи (посао напушта тачно када откуца 00:00). Напиши програм LENKA који учитава тренутно време (у једном реду број сати између 14 и 23, а у наредном реду број минута између 0 и 59) и исписује колико сати и минута је остало до поноћи.

Улаз:	Излаз:	Улаз:	Излаз:	Улаз:	Излаз:	Улаз:	Излаз:
17	6 48	16	7 1	17	7 0	23	0 1
12		59		0		59	

Анализа: Након уноса тренутног времена, наредна поноћ се може представити у облику времена 24:00. Један начин да се уради овај задатак је да се од броја 24:00 одузме учитани број сати и минута, тако што се примени поступак одузимања бројева записаних у бројевној основи 60. Када је број минута у умањенику мањи од броја минута у умањиоцу, потребно је да се изврше позајмице са претходне позиције. Тада се позајмљује један сат и претвара у 60 минута. Ако је број минута у умањиоцу једнак нула, онда се мора извршити враћање позајмице, тако што се резултујући број сати увећа за 1, а резултујући број минута постави на нула.

Програмски језик C++

```
        #include <iostream>
        using namespace std;
        int main()
   {
    int sat, minut, preostaliSati, preostaliMinuti;
    cin >> sat >> minut;
    preostaliSati=24-sat-1;
    preostaliMinuti=60-minut;
    if(preostaliMinuti==60)
    {
        preostaliSati++;
        preostaliMinuti=0;
    }
    cout << preostaliSati<< " "<< preostaliMinuti << endl;
    return 0;
}
```

Програмски језик C

```
        #include <stdio.h>
        int main()
        {
            int sat, minut, preostaliSati, preostaliMinuti;
            scanf("%d%d", &sat, &minut);
            preostaliSati=24-sat-1;
            preostaliMinuti=60-minut;
            if(preostaliMinuti==60)
            {
                preostaliSati++;
                preostaliMinuti=0;
            }
            printf ("%d %d", preostaliSati, preostaliMinuti);
            return 0;
        }
```

Програмски језик Python

```
         sati=int(input())
         minuti=int(input())
         #preostalo vreme u minutima
         ostalo = (60 - minuti) + (24 - sati - 1) * 60
         print ('%d %d' % (ostalo // 60, ostalo % 60))
```

Програмски језик Java

```
         import java.util.*;
```

```
public class Main
{
    public static void main(String[] args)
    {
        Scanner ulaz = new Scanner(System.in);
         int sat = ulaz.nextInt();
        int minut = ulaz.nextInt();
         int preostaliSati, preostaliMinuti;
        preostaliSati=24-sat-1;
        preostaliMinuti=60-minut;
         if(preostaliMinuti==60)
        {
            preostaliSati++;
            preostaliMinuti=0;
        }

        System.out.println(preostaliSati + " " + preostaliMinuti);
    }
}
```

Програмски језик Pascal

```
program zad2;
 var sat, minut, preostaliSati, preostaliMinuti:integer;
 begin
  readln(sat); readln(minut);
  preostaliSati:=24-sat-1;
  preostaliMinuti:=60-minut;
     if(preostaliMinuti=60) then
     begin
         preostaliSati:=preostaliSati+1;
         preostaliMinuti:=0;
     end;
  writeln(preostaliSati, ' ', preostaliMinuti);
 end.
```

Програмски језик C#

```
using System;
class Program
{
    static void Main(string[] args)
    {
        int sati, minuti, ostalo;
        sati = int.Parse(Console.ReadLine());
        minuti = int.Parse(Console.ReadLine());
       //preostalo vreme do ponoci u minutima
        ostalo = (60 - minuti) + (24 - sati - 1) * 60;
        //ispis preostalih sati i preostalih minuta do ponoci
        Console.WriteLine(ostalo/60 + " " + ostalo%60);
    }
}
```

Најчешће грешке: Поједини такмичари су рачунали преостале часове до поноћи тако што

од 23 часа одузму текући сат, али су забуном лоше прерачунали преостале минуте када је текући број минута једнак 0. Та идеја је могла да донесе поене на свим тест примерима, изузев првог тест примера.

2. Написати програм AVION који са улаза учитава у прва два реда времена полетања авиона (у првом реду број сати узмеђу 0 и 23, у другом реду број минута између 0 и 59), а у следећа два реда време слетања авиона (у трећем реду број сати узмеђу 0 и 23, у четвртом реду број минута између 0 и 59) и исписује трајање лета у часовима и минутима. Претпоставити да су полетање и слетање у истом дану, као и да су све вредности исправно унете.

УЛАЗ	**ИЗЛАЗ**
17 12 18 10	0 58

Анализа: (1. начин) Израчунамо колико је секунди протекло од поноћи до почетка полетања и колико секунди је протекло од поноћи до почетка полетања. Одузмемо та два броја и добијемо трајање лета у секундама. Потом испишемо трајање лета у целим сатима и целим минутама.

(2. начин) Израчунамо колико је минута протекло од поноћи до почетка полетања и колико минута је протекло од поноћи до почетка полетања. Одузмемо та два броја и добијемо трајање лета у секундама. Потом испишемо трајање лета у целим сатима и целим минутама.

Програмски језик C++

```
#include <iostream>
using namespace std;
int main()
{
    int poletanjeSat, poletanjeMinut, sletanjeSat, sletanjeMinut,
poletanje, sletanje, trajanjeLeta;
    cin >> poletanjeSat >> poletanjeMinut;
    cin >> sletanjeSat >> sletanjeMinut;
    //pretvoriti vreme poletanja u sekunde
    poletanje=poletanjeSat*3600+poletanjeMinut*60;
    //pretvoriti vreme sletanja u sekunde
    sletanje=sletanjeSat*3600+sletanjeMinut*60;
//izracunati trajanje leta u sekundama
    trajanjeLeta=sletanje-poletanje;
    //izdvajamo broj sati i broj minuta trajanja leta
    int brojSati = trajanjeLeta/3600;
    int brojMinuta = (trajanjeLeta %3600)/60;
    cout << brojSati << " "<< brojMinuta << endl;
    return 0;
}
```

Програмски језик C

```
#include <stdio.h>
int main()
{
    int poletanjeSat, poletanjeMinut, sletanjeSat, sletanjeMinut,
poletanje, sletanje, trajanjeLeta;
    scanf("%d%d", &poletanjeSat, &poletanjeMinut);
    scanf("%d%d", &sletanjeSat, &sletanjeMinut);
```

```
    //pretvoriti vreme poletanja u sekunde
    poletanje=poletanjeSat*3600+poletanjeMinut*60;
    //pretvoriti vreme sletanja u sekunde
    sletanje=sletanjeSat*3600+sletanjeMinut*60;
//izracunati trajanje leta u sekundama
    trajanjeLeta=sletanje-poletanje;
    //izdvajamo broj sati i broj minuta trajanja leta
    int brojSati = trajanjeLeta/3600;
    int brojMinuta = (trajanjeLeta %3600)/60;
    printf("%d %d\n", brojSati, brojMinuta);
    return 0;
}
```

Програмски језик Python

```
poletanjeSat=int(input())
poletanjeMinut=int(input())
sletanjeSat=int(input())
sletanjeMinut=int(input())
minutiPoletanja=poletanjeSat*60+poletanjeMinut
minutiSletanja=sletanjeSat*60+sletanjeMinut
#II nacin - izracunati trajanje leta u minutama
trajanjeLeta=minutiSletanja-minutiPoletanja

sati=trajanjeLeta//60
minuti=trajanjeLeta %60
print(sati, minuti)
```

Програмски језик Java

```
import java.util.*;
public class Main
{
    public static void main(String[] args)
    {
        Scanner ulaz = new Scanner(System.in);
        int poletanjeSat = ulaz.nextInt();
        int poletanjeMinut = ulaz.nextInt();

        int sletanjeSat = ulaz.nextInt();
        int sletanjeMinut = ulaz.nextInt();
        int poletanjeUMinutama = (poletanjeSat * 60) +
poletanjeMinut;
        int sletanjeUMinutama = (sletanjeSat * 60) + sletanjeMinut;
        int trajanjeLeta = sletanjeUMinutama - poletanjeUMinutama;

        System.out.println(trajanjeLeta/60 + " " + trajanjeLeta%60);
    }
}
```

Програмски језик Pascal

```
program zad3;
var poletanjeSat, poletanjeMinut, sletanjeSat, sletanjeMinut:integer;
     minutiPoletanja, minutiSletanja, trajanjeLeta, sati, minuti
:integer;
```

```
begin
 readln(poletanjeSat); readln(poletanjeMinut);
 readln(sletanjeSat); readln(sletanjeMinut);
 minutiPoletanja:=poletanjeSat*60+poletanjeMinut;
 minutiSletanja:=sletanjeSat*60+sletanjeMinut;
 trajanjeLeta:=minutiSletanja-minutiPoletanja;
 sati:=trajanjeLeta div 60;
 minuti:=trajanjeLeta mod 60;
 writeln(sati, ' ', minuti);
end.
```

Програмски језик C#

```
using System;
class Program
{
    static void Main(string[] args)
    {
      int poletanjeSat = int.Parse(Console.ReadLine());
        int poletanjeMinut = int.Parse(Console.ReadLine());
        int sletanjeSat = int.Parse(Console.ReadLine());
        int sletanjeMinut = int.Parse(Console.ReadLine());
  int poletanjeUMinutama = (poletanjeSat * 60) + poletanjeMinut;
  int sletanjeUMinutama = (sletanjeSat * 60) + sletanjeMinut;
int trajanjeLeta = sletanjeUMinutama - poletanjeUMinutama;
        //ispis trajanje leta u celobrojnim satima i minutima
  Console.WriteLine(trajanjeLeta/60 + " " + trajanjeLeta%60);
    }}
```

Најчешће грешке: Поједини такмичари су грешили при рачунање у бројевном систему са основом 60. Наиме, трајање лета израчунава се тако што се од времена доласка одузме време поласка. Задатак, дакле, захтева израчунавање разлике између два временска тренутка за која знамо да су у оквиру једног дана. Како су оба временска тренутка записана у бројевној основи 60, могуће је применити алгоритам одузимања бројева датих у позиционом запису у датој основи.

Одузимање креће од позиција најмање тежине (у овом случају то су минути), при чему се врши позајмица са претходне позиције ако је то потребно (ако је број на датој позицији у умањиоцу већи од броја на датој позицији у умањенику). Позајмице је могуће вршити током одузимања. Али, могуће је позајмице извршити и одмах на почетку, у фази претварања оба временска тренутка у минуте, а тек онда кренути са одузимањем (које се онда врши у бројевној основи 10, јер више нема потребе за позајмицама).

3. Напиши програм MAKSTROCIFREN који учитава троцифрен број, а на стандардни излаз исписује највећи троцифрен број који се састоји од истих цифара као и број на улазу.

Улаз:	Излаз:	Улаз:	Излаз:	Улаз:	Излаз:
123	321	174	741	505	550

Анализа: За учитани број, израчунамо цифре стотина, десетица и јединица коришћењем количника и остатка при целобројном дељењу са 100 и са 10. Потом нађемо цифру која је највећа по вредности. Затим нађемо цифру која је најмања по вредности. Преостала цифра је средња по вредности. Потом формирамо број код кога највећа цифра представља цифру стотина, а најмања цифра представља цифру јединица. Испишемо формирани број.

Програмски језик C++

```
#include <iostream>
```

```
#include <algorithm>
using namespace std;
int main()
{
    int broj, stotine, desetice, jedinice, najvecaCifra,
najmanjaCifra, sredina;
    cin >> broj;
    stotine=broj/100;
    desetice=(broj%100)/10;
    jedinice=broj%10;
    najvecaCifra= max(max(stotine,desetice), jedinice);
    najmanjaCifra= min(min(stotine,desetice), jedinice);
    sredina= (stotine+desetice+jedinice) -
(najvecaCifra+najmanjaCifra);
    cout << 100*najvecaCifra+10*sredina+najmanjaCifra;
    return 0;
}
```

Програмски језик Python

```
broj=int(input())
stotine=broj//100
desetice=(broj%100)//10
jedinice=broj%10
najvecaCifra= max(stotine,desetice, jedinice)
najmanjaCifra= min(stotine,desetice, jedinice)
sredina= (stotine+desetice+jedinice) - (najvecaCifra+najmanjaCifra)
print(100*najvecaCifra+10*sredina+najmanjaCifra)
```

Најчешће грешке: Поједини такмичари су тачно одредили највећу и најмању цифру, али су погрешно претпоставили да цифра која је средња по вредности је строго мања од најмање од највеће цифре и строго већа од најмање цифре. Видимо по тест примерима да такав програм не даје коректан резултат када број садржи више појава најмање или највеће цифре.
Неки такмичари нису успели да коректно одреде максимум три цифре.
Неки такмичари су погрешили при одређивању минимума три цифре.

4. Напишите програм RAZLIKA, који учитава са улаза троцифрени број и израчунава разлику између највећег и најмањег троцифреног броја, који се записује истим цифрама као и учитани број.

Улаз:	Излаз:	Улаз:	Излаз:	Улаз:	Излаз:
123	198	174	594	505	45

Програмски језик C++

```
#include <iostream>
using namespace std;
int main()
{
    int n,x,y,z;
    cin >> n;       // unos trocifrenog broja
    x= n/100;      // stotine
    y = n/10%10;    // desetice
    z = n%10;       // jedinice
    if(x>y) swap(x,y);  // sortiranje
    if(y>z) swap(y,z);  // tri cifre
```

```
            if(x>y) swap(x,y);  // z, y, x u nerastucem poretku
int najveciBroj = z*100 + y*10 + x; // najveci trocifren broj zyx

            if(x==0) swap(x,y);  // ako je broj sa jednom cifrom 0
            if(x==0) swap(x,z);  // ako je broj sa 2 cifre nula
int najmanjiBroj = x*100 + y*10 + z; // najmanji trocifren broj xyz
            cout << najveciBroj - najmanjiBroj << endl; // razlika
            return 0;
        }
```

5. Пера и Ана раде у истој програмерској фирми. Ако се зна радно време сваког од њих напиши програм INTERVAL који одређује колико су времена провели на послу заједно. Са улаза се учитава време када је Пера дошао на посао, време када је отишао, време када је Ана дошла на посао и време када је она отишла. Свако време задаје се у посебној линији, преко два броја одвојених једним размаком. Резултат исписати у облику броја сати и минута, опет раздвојених једним размаком.

Улаз:	Излаз:
8 50	5 5
16 40	
9 20	
14 25	

Анализа: Пресек два временска интервала. Први временски интервал представља Перин боравак на послу, а други временски интервал представља Анин боравак на послу.

Почетак евентуалног пресека је већи од два времена доласка на посао тј. `apresek=max(ПераДолазакНаПосао,АнаДолазакНаПосао))`

Крај евентуалног пресека је мањи од од два времена одласка са посла тј. `bpresek=min(ПераОдлазакСаПосла,АнаОдлазакСаПосла))`.

Пресек постоји ако и само ако је `bpresek>apresek`.

Најчешће грешке: Грешке у одређивању пресека два временска интервала. Неки ученици су уместо пресека два временска интервала заправо одређивали покривач два временска интервала и на тај начин уместо проблема пресека скупова решавали проблем уније скупова.

Програмски језик C++

```
        #include <iostream>
        using namespace std;
        int main ()
        {
           int   pera1sat,   pera1minut,   pera2sat,   pera2minut,   ana1sat,
        ana1minut, ana2sat, ana2minut;
           int peradolazak,peraodlazak,anadolazak,anaodlazak,dolazak,odlazak;
           cin >>pera1sat>>pera1minut; //Pera, dolazak na posao
           cin >>pera2sat>>pera2minut; //Pera, odlazak sa posla
           cin >>ana1sat>>ana1minut;
           cin >>ana2sat>>ana2minut;
           peradolazak=pera1sat*60+pera1minut;
           peraodlazak=pera2sat*60+pera2minut;
           anadolazak=ana1sat*60+ana1minut;
           anaodlazak=ana2sat*60+ana2minut;
           dolazak=max(peradolazak, anadolazak);
           odlazak=min(peraodlazak,anaodlazak);
           if (dolazak>odlazak) cout<<0<<" "<<0;
           else cout<<(odlazak-dolazak)/60<< " "<<(odlazak-dolazak)%60;
```

```
   return 0;
}
```

Програмски језик Python

```
pera1s, pera1m = map (int, input().split())
#pera, pocetak radnog vremena
 pera2s, pera2m = map (int, input().split())
#pera, kraj radnog vremena
ana1s,      ana1m      =      map      (int,      input().split())
#ana, pocetak radnog vremena
ana2s,      ana2m      =      map      (int,      input().split())
#ana, kraj radnog vremena

pera_dolazak                    =                    pera1s*60+pera1m
#pera, pocetak radnog vremena (minuti u odnosu na ponoc)
pera_odlazak                    =                    pera2s*60+pera2m
#pera, kraj radnog vremena (minuti u odnosu na ponoc)
ana_dolazak = ana1s*60+ana1m
ana_odlazak = ana2s*60+ana2m

pocetak               =               max(pera_dolazak,ana_dolazak)
#pocetak eventualnog preseka dva intervala
kraj                  =                  min(pera_odlazak,ana_odlazak)
#kraj eventualnog preseka dva intervala

if pocetak > kraj:
    zajedno = 0
else:
    zajedno = kraj - pocetak

presek_sati = zajedno // 60
presek_minuta = zajedno % 60
print (presek_sati,presek_minuta)
```

6. Дат је скуп који садржи N природних бројева. Нађи највећи подскуп такав да за било који пар елемената A и B из тог подскупа или A дели B или B дели A. У првом реду стандардног улаза дат је цео број N ($1 \le N \le 2000$). У другој линији дато је N чланова скупа (све вредности су између 1 и 10^9). На стандарни излаз исписати један цео број који представља број чланова жељеног скупа.

Примери

Улаз	Излаз	Појашњење
5 9 36 17 108 3	4	Решење са 4 елемента је (9,36,108,3).
9 18 20 6 200 4 36 108 40 2	5	Решење са 5 елемената је (20,200,4,40,2).
10 16 108 8 360 6 756 18 2 36 34	6	Решење са 6 елемената је (108,6,756,18,2,36).

Решење:

Ако број ***A*** дели број ***B*** и број ***B*** дели број ***C*** онда број ***A*** дели број ***C***. (Докажите за вежбу) Транзитивност нам омогућава да извршимо једноставну проверу да видими да ли је неки подскуп ваљан у смислу ограничења описаног у задатку: сортирамо све бројеве у подскупу и проверимо да ли важи дељивост за узастопне бројеве. То заправо значи да након сортирања морамо да нађемо најдужи подниз ***S*** тако да члан ***S[i]*** дели ***S[i+1]***. Користимо ДП технику да решимо задатак као што смо учили да решавамо сличан проблем налажења дужине највећег заједничког подниза.

Програмски језик C++

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

int main() {
    int n; cin >> n;
    vector<int> skup(n), resenje(n, 0);
    for (int i = 0; i < n; i += 1) {
        cin >> skup[i];
    }

    sort(skup.begin(), skup.end());
    for (int i = 0; i < n; i += 1) {
        int najbolje_resenje = 0;
        for (int j = 0; j < i; j += 1) {
            if (skup[i] % skup[j] == 0 and najbolje_resenje < resenje[j]) {
                najbolje_resenje = resenje[j];
            }
        }
        resenje[i] = najbolje_resenje + 1;
    }

    cout << *max_element(resenje.begin(), resenje.end());
    return 0;
}
```

Програмски језик Python

```
n = int(input())

skup = list(reversed(sorted([int(x) for x in input().split()])))
dp = [0 for i in range(n)]
for i in range(n):
   for j in range(i):
      if skup[j] % skup[i] == 0 and dp[j] > dp[i]:
         dp[i] = dp[j]
   dp[i] += 1
print(max(dp))
```

7. Посматрајмо низ бројева чији су прости чиниоци само 2, 3 и 5 (сваки може да се јави нула и више пута). То су бројеви 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, 18, ... Напиши програм који одређује n-ти члан овог низа (бројање креће од 0). Низ бројева зовемо Вивалдијев низ. Са стандардног улаза се учитава број n ($0 \leq n \leq 10000$). На стандардни излаз исписати тражени n-ти члан низа.

Улаз	Излаз
500	944784

Анализа: Користићемо класичну Python листу због уштеде времена, јер морамо да итерирамо кроз њу. У C++ би користили deque. тј. Queue у језику C#.
Процена: наша листа садржи 10000 64-битних вредости. Сваки елемент овог низа добија се множењем неког мањег елемента са 2, 3 или са 5. Замислимо да је низ 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, 18, ... већ конструисан. Ако сваки елемент помножимо са 2, добијамо низ 2, 4, 6, 8, 10, 12, 16, 18, ... Ако сваки елемент помножимо са 3, добијамо низ 3, 6, 9, 12, 15, 18, 24, 27, 30, ... Ако сваки елемент помножимо са 5, добијамо низ 5, 10, 15, 20, 25, 30, 40, ... Приметимо да се обједињавањем ова три низа добија низ који тражимо. Ово инсиприше следећи поступак. Чувамо 3 реда у којима се налазе елементи три низа која обједињавамо. На почетку у први постављамо само број 2, у други број 3, а у трећи број 5. Узимамо минимални елемент та три низа и уклањамо га са почетка сва три низа. Множимо га редом, са 2, 3 и 5 и додајемо резултате на крај одговарајућих редова. Поступак настављамо све док са почетка редова не издвојимо n елемената. Алгоритам ће имати сложеност O(n) зато што имамо два једно-димензиона итератора.

```
vivaldi = [1]

a = b = c = 0
i=0
n=int(input())
while i<n:
    sledeci = min(2*vivaldi[a], 3*vivaldi[b], 5*vivaldi[c])
    vivaldi.append(sledeci)
    if sledeci == 2*vivaldi[a]: a += 1
    if sledeci == 3*vivaldi[b]: b += 1
    if sledeci == 5*vivaldi[c]: c += 1
    i+=1
print (vivaldi[n])
```

Програмски језик C++

```
#include <iostream>
#include <queue>
#include <algorithm>
using namespace std;
int main() {
  int n;
  cin >> n;
  deque<unsigned long long> niz2, niz3, niz5;
  niz2.push_back(2); niz3.push_back(3); niz5.push_back(5);
  unsigned long long vivaldi = 1;
  for (int i = 0; i < n; i++) {
    vivaldi = min({niz2.front(), niz3.front(), niz5.front()});
    niz2.push_back(2*vivaldi);
    niz3.push_back(3*vivaldi);
    niz5.push_back(5*vivaldi);
    while (niz2.front() == vivaldi) niz2.pop_front();
    while (niz3.front() == vivaldi) niz3.pop_front();
    while (niz5.front() == vivaldi) niz5.pop_front();
  }
  cout << vivaldi << endl;
  return 0;
}
```

Програмски језик C#

```
using System.Collections.Generic;
class Program
{
    static void Main(string[] args)
    {
        int n = int.Parse(Console.ReadLine());
        var a2 = new Queue<long>();
        var a3 = new Queue<long>();
        var a5 = new Queue<long>();
        a2.Enqueue(2); a3.Enqueue(3); a5.Enqueue(5);
        long t = 1;
        for (int i = 0; i < n; i++) {
            t = Math.Min(a2.Peek(), Math.Min(a3.Peek(), a5.Peek()));
            a2.Enqueue(2*t);
            a3.Enqueue(3*t);
            a5.Enqueue(5*t);
            while (a2.Peek() == t) a2.Dequeue();
            while (a3.Peek() == t) a3.Dequeue();
            while (a5.Peek() == t) a5.Dequeue();
        }
        Console.WriteLine(t);
    }
}
```

Програмски језик Haskell

```
merge :: Ord a => [a] -> [a] -> [a]
merge [] xs = xs
merge xs [] = xs
merge (x:xs) (y:ys) |  x < y   = x : merge xs (y:ys)
                    |  x > y   = y : merge (x:xs) ys
                    |  x == y  = x : merge xs ys

merge3 :: Ord a => [a] -> [a] -> [a] -> [a]
merge3 a b c = merge (merge a b) c

cinioci235_niz :: [Integer]
cinioci235_niz = 1 : (merge3
                       (map (*2) cinioci235_niz)
                       (map (*3) cinioci235_niz)
                       (map (*5) cinioci235_niz)
                     )

cinioci235 :: Int -> Integer
cinioci235 n = cinioci235_niz !! n
```

8. Са улаза се уносе цели бројеви све док се не дође до краја улаза. Написати програм којим се приказује колико је унето бројева.

Решење 1 - Пајтон	Решење 2 - Ц++	Решење 3 – Ц#

`import sys`
`broj = 0`
`for linija in sys.stdin:`
	`#include <iostream>`
	
	`using namespace std;`
		`using System;`
		
		`class Program`

```
    broj = broj + 1
print(broj)
```

```
int main() {
   int x;
   int br = 0;
   while (cin >> x)
      br++;
   cout << br << endl;
   return 0;
}
```

```
{
    static void Main(string[]
args)
    {
        int br = 0;
        while
(Console.ReadLine() != null)
            br++;
        Console.WriteLine(br);
    }
}
```

9. Творац Python-а се опирао увођењу ламбда оператора односно ламбда функција (јер постоји concept list comprehension): Having both list comprehension and "Filter, map, reduce and lambda" is transgressing the Python motto "There should be one obvious way to solve a problem".

Решење 1

```
# broj polica i broj flasa u gajbici
n = int(input())
k = int(input())
ukupanBrojGajbica = 0
for i in range(n):
   # broj flasa na trenutnoj polici
   f = int(input())
   # ukupan broj gajbica se uvecava za broj
gajbica potreban da se
   # smeste flase na trenutnoj polici
   ukupanBrojGajbica = ukupanBrojGajbica + f //
k
   if (f % k != 0):
      ukupanBrojGajbica = ukupanBrojGajbica + 1

# ispisujemo ukupan broj gajbica
print(ukupanBrojGajbica)
```

Решење 2

```
# broj polica i broj flasa u gajbici
n = int(input())
k = int(input())
# ucitavamo broj flasa na svakoj polici
brojFlasa = (int(input()) for i in range(n))
# izracunavamo broj gajbica potrebnih za flase na
svakoj polici
brojGajbica = map(lambda f: (f + k - 1) // k,
brojFlasa)
# izracunavamo i ispisujemo ukupan broj gajbica
ukupanBrojGajbica = sum(brojGajbica)
print(ukupanBrojGajbica)
```

Мапирање у другим програмским језицима:

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <numeric>
using namespace std;
int main() {
    int n, k;
  cin >> n >> k;
    vector<int> brojFlasa(n);
  for (int i = 0; i < n; i++)
    cin >> brojFlasa[i];

  // izracunavamo broj gajbica potrebnih za flase na svakoj polici
  vector<int> brojGajbica(n);
  transform(begin(brojFlasa), end(brojFlasa), begin(brojGajbica),
          [k](int f) { return (f + k - 1) / k; });

  // izracunavamo ukupan broj gajbica
  int ukupanBrojGajbica =    accumulate(begin(brojGajbica), end(brojGajbica), 0);
  cout << ukupanBrojGajbica << endl;
  return 0;
```

```
}
```

C#

```
using System;
using System.Linq;
class Program
{
    static void Main(string[] args)
    {
        int n = int.Parse(Console.ReadLine());
        int k = int.Parse(Console.ReadLine());
      int[] brojFlasa = new int[n];
      for (int i = 0; i < n; i++)
          brojFlasa[i] = int.Parse(Console.ReadLine());

      // izracunavamo broj gajbica potrebnih za flase na svakoj polici
      var brojGajbica = brojFlasa.Select(f => (f + k - 1) / k);

      // izracunavamo ukupan broj gajbica
      int ukupanBrojGajbica = brojGajbica.Sum();
              Console.WriteLine(ukupanBrojGajbica);
    }
}
```