ОКРУЖНО ТАКМИЧЕЊЕ ИЗ РАЧУНАРСТВА ЗА УЧЕНИКЕ ОСНОВНХ ШКОЛА

БИЛТЕН

ДРЖАВНА КОМИСИЈА ЗА ТАКМИЧЕЊЕ УЧЕНИКА ОСНОВНИХ ШКОЛА

2017/18

18.03. 2018.

ДМС

Садржај:

Окружно такмичење ученика основних школа из рачунарства - пети разред
(18.03. 2018.)

1. [stepeni] Збир унутрашњих углова сваког троугла је 180 степени. Троугао је оштроугли ако су му сва три угла оштра (мања од 90 степени), правоугли ако му је један угао прав (једнак 90 степени), а тупоугли ако му је један угао туп (већи од 90 степени). Напиши програм који учитава величине два угла троугла у степенима (цели бројеви, сваки у посебном реду), одређује да ли је тај троугао оштроугли, правоугли или тупоугли и исписује одговарајући текст (латиницом).

Примери

Улаз:	*Излаз:*	*Улаз:*	*Излаз:*	*Улаз:*	*Излаз:*
30	pravougli	18	tupougli	75	ostrougli
60		18		60	

Програмски језик C++

```
#include <iostream>
#incl ude <cassert>

using namespace std;

int main() {
   // ucitavamo uglove u stepenima i minutima
   int ugao1, ugao2;
   cin >> ugao1 >> ugao2;
   int ugao3 = 180 - (ugao1 + ugao2);
   assert(0 < ugao3 && ugao3 < 180);


   // ako je bar jedan ugao tup, trougao je tupougli
   if (ugao1 > 90 || ugao2 > 90 || ugao3 > 90)
   cout << "tupougli" << endl;
  // u suprotnom, ako je bar jedan ugao prav, trougao je pravougli
  else if (ugao1 == 90 || ugao2 == 90 || ugao3 == 90)
  cout << "pravougli" << endl;
 // u suprotnom nema pravih ni tupih uglova, pa je trougao ostrougli
else
      cout << "ostrougli" << endl;


   return 0;
}
```

Програмски језик Python

```
   # ucitavamo uglove
   ugao1 = int(input())
   ugao2 = int(input())
   ugao3 = 180 - (ugao1 + ugao2)

   # ako je bar jedan ugao tup, trougao je tupougli
   if ugao1 > 90 or ugao2 > 90 or ugao3 > 90:
```

```
   print("tupougli")
# u suprotnom, ako je bar jedan ugao prav, trougao je pravougli
elif ugao1 == 90 or ugao2 == 90 or ugao3 == 90:
   print("pravougli")
# u suprotnom nema pravih ni tupih uglova, pa je trougao
ostrougli else:
   print("ostrougli")
```

2. [iksoks] Мирко је мали програмер који покушава да испрограмира игрицу икс-окс. Близу је да заврши, али му је потребна мала помоћ. Смислио је да корисник мишем одређује квадрат у који ће се његов симбол уписати. Поље за игру се састоји од 9 квадрата (распоређена у три врсте и три колоне) и сваки квадрат је димензије 100 пута 100 пиксела (поље је димензије 300 пута 300 пиксела). Познат је положај пиксела на који је кликнуто мишем. Потребно је одредити редни број квадрата у којем се тај пикселналази. Положај пиксела је одређен редним бројевима (координатама) тог пиксела по хоризонтали и по вертикали, рачунајући од доњег левог угла поља (пиксели се броје од 1 до 300). Квадрати се броје од 1 до 9, врсту по врсту, почевши од доњег левог угла поља навише, како је приказано на слици.

7	8	9
4	5	6
1	2	3

Примери

Улаз:	*Излаз:*	*Улаз:*	*Излаз:*	*Улаз:*	*Излаз:*	*Улаз:*	*Излаз*
1	1	120	8	100	7	101	5
1		280		201		200	

Програмски језик C++

```
#include <iostream>

using namespace std;

int main() {
 // dimenzija kvadrata
 const int a = 100;
 // koordinate piksela
 int x, y;
 cin >> x >> y;
 // redni broj vrste i kolone u kojoj se nalazi piksel
 int k = (x - 1) / a, v = (y - 1) / a;
 // redni broj kvadrata
 int kvadrat = 3 * v + k + 1;
 cout << kvadrat << endl; return 0;
}
```

```
#include <iostream>

using namespace std;

int main() {
   // koordinate piksela
   int x, y;
   cin >> x >> y;

   // redni broj kvadrata
   int kvadrat;

   // analiziramo sve slucajeve
   if (1 <= x && x <= 100 && 1 <= y && y <= 100)
      kvadrat = 1;
   if (101 <= x && x <= 200 && 1 <= y && y <= 100)
      kvadrat = 2;
   if (201 <= x && x <= 300 && 1 <= y && y <= 100)
      kvadrat = 3;
   if (1 <= x && x <= 100 && 101 <= y && y <= 200)
      kvadrat = 4;
   if (101 <= x && x <= 200 && 101 <= y && y <= 200)
      kvadrat = 5;
   if (201 <= x && x <= 300 && 101 <= y && y <= 200)
      kvadrat = 6;
   if (1 <= x && x <= 100 && 201 <= y && y <= 300)
      kvadrat = 7;
   if (101 <= x && x <= 200 && 201 <= y && y <= 300)
      kvadrat = 8;
   if (201 <= x && x <= 300 && 201 <= y && y <= 300)
      kvadrat = 9;

 // ispisujemo resenje
   cout << kvadrat << endl;
   return 0;
}
```

Програмски језик Python

```
# dimenzija kvadrata
a = 100
# koordinate piksela
x = int(input())
y = int(input())
# redni broj vrste i kolone u kojoj se nalazi piksel
k = (x - 1) // a
v = (y - 1) // a
# redni broj kvadrata
kvadrat = 3 * v + k + 1
print(kvadrat)

# koordinate piksela
x = int(input())
y = int(input())

# analiziramo sve slucajeve
if 1 <= x and x <= 100 and 1 <= y and y <= 100:
      kvadrat = 1
if 101 <= x and x <= 200 and 1 <= y and y <= 100:
```

```
        kvadrat = 2;
    if 201 <= x and x <= 300 and 1 <= y and y <= 100:
        kvadrat = 3;
    if 1 <= x and x <= 100 and 101 <= y and y <= 200:
        kvadrat = 4;
    if 101 <= x and x <= 200 and 101 <= y and y <= 200:
        kvadrat = 5;
    if 201 <= x and x <= 300 and 101 <= y and y <= 200:
        kvadrat = 6;
    if 1 <= x and x <= 100 and 201 <= y and y <= 300:
        kvadrat = 7;
    if 101 <= x and x <= 200 and 201 <= y and y <= 300:
        kvadrat = 8;
    if 201 <= x and x <= 300 and 201 <= y and y <= 300:
        kvadrat = 9;

    # ispisujemo resenje
    print(kvadrat)
```

3. [dzudo] На једном турниру џудисти се такмиче у три категорије: до 50 килограма, од 51 до 75 килограма и од 76 килограма навише. Напиши програм који учитава број џудиста једног клуба пријављеног на тај турнир (цео број између 1 и 100), а затим масу сваког од њих (цели бројеви између 40 и 120, сваки у посебном реду) и за сваку категорију редом исписује колико ће се џудиста тог клуба борити у тој категорији.

Примери

Улаз:	*Излаз:*
5 48 51 73 82 50	2 2 1

Објашњење:

у категорији до 50 килограма боре се такмичари који имају 48 и 50 килограма
у категорији од 51 до 75 килограма боре се такмичари који имају 51 и 73 килограма
у категорији од 76 килограма навише бориће се такмичар који има 82 килограма

Програмски језик C++

```
#include <iostream>

using namespace std;

int main() {
  // broj dzudista u raznim kategorijama
  int broj_do_50 = 0;
  int broj_od_51_do_75 = 0;
  int broj_od_76 = 0;
  // ukupan broj dzudista
```

```
  int n;
  cin >> n;
  for (int i = 0; i < n; i++) {
    int tezina;
    cin >> tezina;
    if (tezina <= 50)
      broj_do_50++;
    else if (tezina <= 75)
      broj_od_51_do_75++;
    else
      broj_od_76++;
  }


  cout << broj_do_50 << endl;
  cout << broj_od_51_do_75 << endl;
  cout << broj_od_76 << endl;


  return 0;
}
```

Програмски језик Python

```
# broj dzudista u raznim kategorijama
broj_do_50 = 0
broj_od_51_do_75 = 0
broj_od_76 = 0
# ukupan broj dzudista
n = int(input())
for i in range(n):
    tezina = int(input())
    if tezina <= 50:
        broj_do_50 += 1
    elif tezina <= 75:
        broj_od_51_do_75 += 1
    else:
        broj_od_76 += 1
print(broj_do_50)
print(broj_od_51_do_75)
print(broj_od_76)
```

4. [loto] У једном одељењу ученици су одлучили да у склопу новогодишње приредбе организују мало извлачење игре лото. Лото се игра тако што се из бубња у коме се налази n куглица обележених бројевима од 1 до n извлаче три куглице (извлачи се једна по једна куглица, а извучене куглице се не враћају у бубањ). Када се све куглице извуку, бројеви који пишу на њима се поређају од најмањег до највећег. На пример, ако у бубњу има пет куглица, могуће је да се прво извуче куглица 4, затим 1 и онда 2 - то извлачење се представља тројком бојева 1 2 4 (јер су након извлачења куглице поређане по величини). Напиши програм који на стандардни излаз исписује све могућности (комбинације) које могу бити извучене (тј. све тројке бројева које их представљају). Са стандардног улаза се учитава само број n (важи 3 ≤ n ≤ 9). Свака могућност се приказује у посебном реду, три броја која је представљају су увек уређена од најмањег до највећег и раздвојена размаком, а

могућности се приказују лексикографским редом (што значи да би троцифрени бројеви који би се добили када би се размаци обрисали били поређани од најмањег до највећег).

Примери

Улаз:	*Излаз:*	*Улаз:*	*Излаз:*
5	1 2 3 1 2 4 1 2 5 1 3 4 1 3 5 1 4 5 2 3 4 2 3 5 2 4 5 3 4 5	4	1 2 3 1 2 4 1 3 4 2 3 4

Програмски језик C++

```
#include <iostream>

using namespace std;

int main() {
  int n;
  cin >> n;
  for (int b1 = 1; b1 <= n-2; b1++)
    for (int b2 = b1 + 1; b2 <= n-1; b2++)
      for (int b3 = b2 + 1; b3 <= n; b3++)
         cout << b1 << " " << b2 << " " << b3 << endl;
  return 0;
}
```

Програмски језик Python

```
n = int(input())
for b1 in range(1, (n-2)+1):
    for b2 in range(b1+1, (n-1)+1):
        for b3 in range(b2+1, n+1):
print(b1, b2, b3)
```

Окружно такмичење ученика основних школа из рачунарства - шести разред

1. [uglovi] Збир унутрашњих углова сваког троугла је 180 степени. Троугао називамо оштроуглим ако су му сва три угла оштра, правоуглим ако му је један угао прав, а тупоуглим ако му је један угао туп. Напиши програм који на основу величине два угла троугла у степенима и минутима одређује да ли је тај троугао оштроугли, правоугли или тупоугли. Са стандардног улаза се учитавају четири цела броја: број степени и број минута сваког од два угла (сваки број у посебном реду), а на стандардни излаз треба исписати врсту троугла латиницом.

Примери

Улаз:	Излаз:	Улаз:	Излаз:	Улаз:	Излаз:
29	pravougli	17	tupougli	75	ostrougli
30		39		0	
60		18		60	
30		45		0	

Програмски језик C++

```
#include <iostream>
#include <cassert>

using namespace std;

// pretvara ugao dat u stepenima(s) i minutima (m) u
// ugao samo u minutima
int ugao(int s, int m) {
  return s * 60  + m;
}


int main() {
  // ucitavamo uglove u stepenima i minutima
  int ugao1_s, ugao1_m, ugao2_s, ugao2_m, ugao3_s, ugao3_m;
  cin >> ugao1_s >> ugao1_m;
  cin >> ugao2_s >> ugao2_m;


  // pretvaramo ih u uglove date samo u minutima
  int ugao1 = ugao(ugao1_s, ugao1_m);
  int ugao2 = ugao(ugao2_s, ugao2_m);
  int ugao3 = ugao(180, 0) - (ugao1 + ugao2);


  ugao3_s = ugao3 / 60; ugao3_m = ugao3 % 60;


  // prav ugao u minutima
  int pravUgao = ugao(90, 0);
  // ako je bar jedan ugao tup, trougao je tupougli
  if (ugao1 > pravUgao || ugao2 > pravUgao || ugao3 > pravUgao)
    cout << "tupougli" << endl;
  // u suprotnom, ako je bar jedan ugao prav, trougao je pravougli
    else if (ugao1 == pravUgao || ugao2 == pravUgao || ugao3 ==
  pravUgao)
```

```
    cout << "pravougli" << endl;
  // u suprotnom nema pravih ni tupih uglova, pa je trougao ostrougli
  else
    cout << "ostrougli" << endl;

  return 0;
}
```

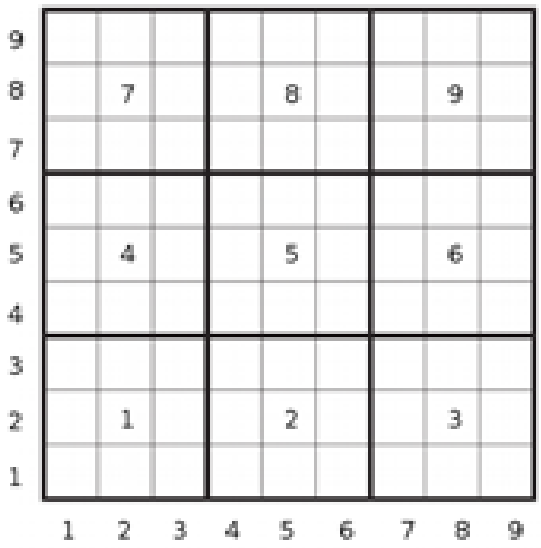
Програмски језик Python

```
# pretvara ugao dat u stepenima(s) i minutima (m) u
# ugao samo u minutima
def ugao(s, m):
    return s * 60 + m


# ucitavamo uglove u stepenima i minutima
ugao1_s = int(input())
ugao1_m = int(input())
ugao2_s = int(input())
ugao2_m = int(input())
# pretvaramo ih u uglove date samo u minutima
ugao1 = ugao(ugao1_s, ugao1_m)
ugao2 = ugao(ugao2_s, ugao2_m)
ugao3 = ugao(180, 0) - (ugao1 + ugao2)


# prav ugao u minutima
pravUgao = ugao(90, 0)
# ako je bar jedan ugao tup, trougao je tupougli
if ugao1 > pravUgao or ugao2 > pravUgao or ugao3 > pravUgao:
    print("tupougli")
# u suprotnom, ako je bar jedan ugao prav, trougao je pravougli
elif ugao1 == pravUgao or ugao2 == pravUgao or ugao3 == pravUgao:
    print("pravougli")
# u suprotnom nema pravih ni tupih uglova, pa je trougao ostrougli
else:
     print("ostrougli")
```

2. [sudoku] Мирко је програмер који покушава да испрограмира игрицу судоку. Близу је да заврши, али му је потребна мала помоћ. Смислио је да корисник мишем бира квадрат у који ће уписати број. Поље се састоји од 81 квадратића, који су распоређени у 9 хоризонталних врста, 9 вертикалних колона и 9 већих квадрата (као на слици). Сваки квадратић је димензије 30 пута 30 пиксела (укупно поље је димензије 270 пута 270 пиксела). Познат је положај пиксела на који је кликнуто мишем. Положај је одређен редним бројевима (координатама) тог пиксела по хоризонтали и по вертикали, рачунајући од доњег левог угла поља (пиксели се и по хоризонтали и по вертикали броје од 1 до 270). Потребно је исписати редни број врсте, колоне и већег квадрата у којем се налази пиксел на који је кликнуто (врсте се броје од 1 до 9 одоздо навише, колоне од 1 до 9 слева надесно, а квадрати по врстама од доњег левог угла, како је обележено на слици), сваки број у посебном реду.



Примери

Улаз:	*Излаз:*	*Улаз:*	*Излаз:*	*Улаз:*	*Излаз:*
128	8	180	6	181	7
230	5	180	6	181	7
	8		5		9

Програмски језик C++

```
#include <iostream>

using namespace std;

int main() {
  // dimenzije kvadratica
  const int a = 30;
  // koordinate piksela
  int x, y;
  cin >> x >> y;
  // vrsta i kolona u kojoj se nalazi kvadratic (na polju 9x9), brojano od 0
  int kolona = (x - 1) / a;
  int vrsta = (y - 1) / a;
  // vrsta i kolona u kojoj se nalazi kvadrat (na polju 3x3), brojano od 0
  int K = kolona / 3;
  int V = vrsta / 3;
  // redni broj kvadrata, brojano od 0
  int kvadrat = V * 3 + K;
  // ispis resenja (brojano od 1)
  cout << vrsta  + 1  << endl;
  cout << kolona  + 1 << endl;
  cout << kvadrat + 1 << endl;
  return 0;
}
```

Програмски језик Python

```
# dimenzije kvadratica
a = 30
# koordinate piksela
x = int(input())
y = int(input())
# vrsta i kolona u kojoj se nalazi kvadratic (na polju 9x9), brojano od 0
kolona = (x - 1) // a
vrsta = (y - 1) // a
# vrsta i kolona u kojoj se nalazi kvadrat (na polju 3x3), brojano od 0
K = kolona // 3
V = vrsta // 3;
# redni broj kvadrata, brojano od 0
kvadrat = V * 3 + K
# ispis resenja (brojano od 1)
print(vrsta + 1)
print(kolona + 1)
print(kvadrat + 1)
```

3. [kartice] Пера се игра са картама на којима пишу природни бројеви између 1 и 10. У сваком дељењу је карте које је добио желео да сложи од најмање до највеће (ако има истих оне се налазе једна уз другу). Напиши програм који израчунава колико пута је Пера погрешио и није сложио карте како је желео. Са стандардног улаза се учитава прво број n (1 ≤ n ≤ 100) - број дељења које је Пера играо, затим број k (2 ≤ k ≤ 10) - број карата у сваком дељењу, а затим за свако дељење по k карата наведених онако како их је Пера сложио. Сваки број је наведен у засебном реду. На стандардни излаз исписати број дељења у којима Пера није добро сложио карте.

Пример

Улаз:	*Излаз:*	*Објашњење:*
2 3 1 2 2 2 4 3	1	Учитане су две поделе са по три карте. Карте у подели 1 2 2 су исправно сложене, а у подели 2 4 3 нису.

Програмски језик C++

```
#include <iostream>
#include <algorithm>

using namespace std;

int main() {
  int brojLosih = 0;
  int n, k;
  cin >> n >> k;
  for (int i = 0; i < n; i++) {
    vector<int> karte(k);
    for (int j = 0; j < k; j++)
```

```
      cin >> karte[j];
    if (!is_sorted(begin(karte), end(karte)))
      brojLosih++;
  }
  cout << brojLosih << endl;
  return 0;
}
```

Програмски језик Python

```
brojLosih = 0
n = int(input())
k = int(input())
for i in range(n):
  karte = [int(input()) for j in range(k)]
  if (not(all(karte[j] <= karte[j+1] for j in range(k-1)))):
      brojLosih += 1
print(brojLosih)
```

4. [prag] Државна комисија треба да одреди праг за пролазак такмичара са окружног на државно такмичење. Пошто је информатика постала обавезан предмет у основним школама, број такмичара је јако велики. Администраторку Мају која одржава табелу са резултатима стално питају који би број такмичара прошао даље када би праг пролазности био толико и толико поена (даље се пласирају сви ученици чији је број поена већи или једнак прагу). Одлучила је да напише програм који даје одговор на та питања. Са стандардног улаза учитава се број такмичара n (1 ≤ n ≤ 50000), а затим и поени такмичара (природни бројеви), задати у сортираном редоследу од највећег до најмањег и раздвојени размацима. Након тога се учитава број m (1 ≤ m ≤ 50000) који представља број питања на која Маја треба да одговори, а затим и m бројева раздвојених размацима за које је потребно дати одговор колико би се такмичара пласирало када би се тај број узео за праг. На стандардни излаз исписати тражене бројеве такмичара који су се пласирали, у посебном реду за сваки праг.

Примери

Улаз:	*Излаз:*	*Објашњење:*
5	0	ако је праг 95 поена, нико се није
89 73 73 56 23	4	ако је праг 50 поена, пласирали су
4	3	ако је праг 95 поена, нико се није
95 50 70 5	5	ако је праг 50 поена, пласирали су

Напомена: У бар 15 од 20 тест примера, бројеви m и n ће бити мањи од 200.

Програмски језик C++

```
#include <iostream>

#include <vector>

using namespace std;
```

```
int main() {
int n;
cin >> n;
vector<int> poeni(n);
for (int i = n-1; i >= 0; i--)
  cin >> poeni[i];

int m; cin >> m;
for (int i = 0; i < m; i++) {
int prag; cin >> prag;
    int broj = distance(lower_bound(begin(poeni), end(poeni), prag),
    end(poeni)); cout << broj << endl;
  }


  return 0;
}



#include <iostream>
#include <vector>


using namespace std;


int prvi_veci_ili_jednak(const vector<int>& a, int x)
  { int l = 0, d = a.size()-1;
while (l <= d) {
    int s = l + (d - l) /
    2; if (a[s] < x)
      l = s + 1;
    else
      d = s - 1;
  }
  return d + 1;
}


  int main() {
  int n;
  cin >> n;
  vector<int> poeni(n);
  for (int i = n-1; i >= 0; i--)
  cin >> poeni[i];


  int m;
  cin >> m;
  for (int i = 0; i < m; i++) {
  int prag;
    cin >> prag;
    cout << n - prvi_veci_ili_jednak(poeni, prag) << endl;
  }


  return 0;
}
```

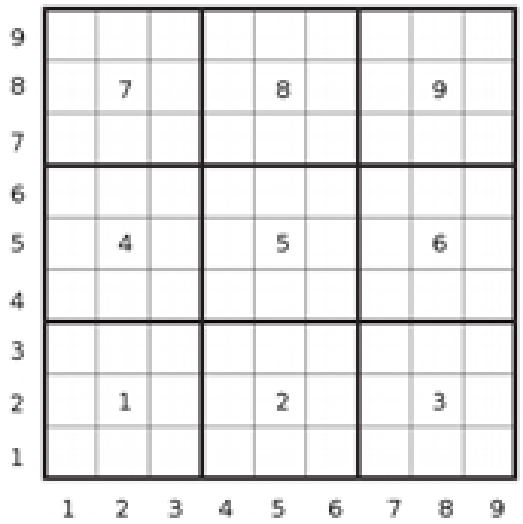
Програмски језик Python

```
import bisect


n = int(input())
poeni = list(map(int, input().split()))
poeni.reverse()
m = int(input())
pragovi = map(int, input().split())
for prag in pragovi:
  print(n - bisect.bisect_left(poeni, prag))
```

Окружно такмичење ученика основних школа из рачунарства - седми разред

1. [sudoku] Мирко је програмер који покушава да испрограмира игрицу судоку. Близу је да заврши, али му је потребна мала помоћ. Смислио је да корисник мишем бира квадрат у који ће уписати број. Поље се састоји од 81 квадратића, који су распоређени у 9 хоризонталних врста, 9 вертикалних колона и 9 већих квадрата (као на слици). Сваки квадратић је димензије 30 пута 30 пиксела (укупно поље је димензије 270 пута 270 пиксела). Познат је положај пиксела на који је кликнуто мишем. Положај је одређен редним бројевима (координатама) тог пиксела по хоризонтали и по вертикали, рачунајући од доњег левог угла поља (пиксели се и по хоризонтали и по вертикали броје од 1 до 270). Потребно је исписати редни број врсте, колоне и већег квадрата у којем се налази пиксел на који је кликнуто (врсте се броје од 1 до 9 одоздо навише, колоне од 1 до 9 слева надесно, а квадрати по врстама од доњег левог угла, како је обележено на слици), сваки број у посебном реду.



Примери

Улаз:	*Излаз:*	*Улаз:*	*Излаз:*	*Улаз:*	*Излаз:*
128	8	180	6	181	7
230	5	180	6	181	7
	8		5		9

Решење: Погледати задатак за 6. разред

2. [parnepar] Пера се игра са картама на којима пишу природни бројеви између 1 и 10. У сваком дељењу је карте које је добио желео да сложи тако да прво иду све карте са парним бројевима, а затим оне са непарним бројевима (могуће је и да је у неком дељењу Пера имао само парне или само непарне карте). Напиши програм који израчунава колико пута је Пера погрешио и није сложио карте онако како је желео. Са стандардног улаза се учитава прво број n (1 ≤ n ≤100) - број дељења које је Пера играо, затим број k (2 ≤ k ≤ 10) - број карата у сваком дељењу, а затим за свако дељење по k карата наведених онако како их је Пера сложио. Сваки број је

наведен у засебном реду. На стандардни излаз исписати број дељења у којима Пера није добро сложио карте.

Примери

Улаз:	*Излаз:*	*Објашњење:*
2 3 2 4 1 2 3 4	1	Учитане су две поделе са по три карте. Карте у подели 2 4 1 су исправно сложене (јер иду прво парне, па непарне), а у подели 2 3 4 нису (јер не иду прво парне, па непарне).

Програмски језик C++

```
#include <iostream>

using namespace std;

int main() {
  int brojLosih = 0;
  int n, k;
  cin >> n >> k;
  for (int i = 0; i < n; i++) {
    bool uRedu = true;
    bool bilaNeparna = false;
    for (int j = 0; j < k; j++) {
      int karta;
      cin >> karta;
      if (karta % 2 == 1)
      bilaNeparna = true;
      else if (bilaNeparna)
      uRedu = false;
    }
    if (!uRedu)
      brojLosih++;
  }
  cout << brojLosih << endl;
  return 0;
}
```

Програмски језик Python

```
brojLosih = 0
n = int(input())
k = int(input())
for i in range(n):
  uRedu = True
  bilaNeparna = False
  for j in range(k):
    karta = int(input())
    if karta % 2 == 1:
        bilaNeparna = True
    elif bilaNeparna:
```

```
      uRedu = False
  if not(uRedu):
    brojLosih += 1
  print(brojLosih)
```

3. [numeracija] Књига има n страна. Колико је цифара употребљено у њиховој нумерацији (она, наравно, креће од 1)? Са стандардног улаза се учитава број n ($1 \leq n \leq 2 \cdot 10^8$). На стандардни излаз исписати тражени број цифара.

Примери

Улаз:	*Излаз:*	*Улаз:*	*Излаз:*
10	11	785	2247

Објашњење: Нумерација 10 страна се врши бројевима 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 и у њима укупно има 11 цифара.

Програмски језик C++

```
#include <iostream>

using namespace std;

int main() {

   int n;
   cin >> n;
   int broj = 0;
   // krecemo od jednocifrenih brojeva
   int s = 10; // interval [s/10, s-1] tj. [1, 9]
   int d = 1;  // broj cifara je 1
   while (s - 1 < n) {
   // dodajemo cifre koriscene za zapis brojeva u intervalu [s/10, s-1]
   broj += ((s - 1) - s/10 + 1) * d;
   // u intervalu [a, b] ima (a - b + 1) brojeva
   // prelazimo na sledeci interval, tj. brojeve sa jednom cifrom vise
   s *= 10;
   d += 1;
}
// preostali su brojevi u intervalu [s/10, n]
// dodajemo njihove cifre
broj += (n - s/10 + 1) * d;
cout << broj << endl;
return 0;
}
```

Програмски језик Python

```
n = int(input())
broj = 0
# krecemo od jednocifrenih brojeva
s = 10     # interval [s/10, s-1] tj. [1, 9]
d = 1      # broj cifara je 1
while s - 1 < n:
     # dodajemo cifre koriscene za zapis brojeva u intervalu [s/10, s-1]
     broj += ((s - 1) - s//10 + 1) * d
```

```
        # u intervalu [a, b] ima (a - b + 1) brojeva
        # prelazimo na sledeci interval, tj. brojeve sa jednom cifrom vise
        s *= 10
        d += 1

    # preostali su brojevi u intervalu [s/10, n]
    # dodajemo njihove cifre
    broj += (n - s//10 + 1) * d
    print(broj)
```

4. [prag] Државна комисија треба да одреди праг за пролазак такмичара са окружног на државно такмичење. Пошто је информатика постала обавезан предмет у основним школама, број такмичара је јако велики. Администраторку Мају која одржава табелу са резултатима стално питају који би број такмичара прошао даље када би праг пролазности био толико и толико поена (даље се пласирају сви ученици чији је број поена већи или једнак прагу). Одлучила је да напише програм који даје одговор на та питања. Са стандардног улаза учитава се број такмичара n (1 ≤ n ≤ 50000), а затим и поени такмичара (природни бројеви), задати у сортираном редоследу од највећег до најмањег и раздвојени размацима. Након тога се учитава број m (1 ≤ m ≤ 50000) који представља број питања на која Маја треба да одговори, а затим и m бројева раздвојених размацима за које је потребно дати одговор колико би се такмичара пласирало када би се тај број узео за праг. На стандардни излаз исписати тражене бројеве такмичара који су се пласирали, у посебном реду за сваки праг.

Примери

Улаз:	*Излаз:*	*Објашњење:*
5	0	ако је праг 95 поена, нико се није
89 73 73 56 23	4	ако је праг 50 поена, пласирали су
4	3	ако је праг 95 поена, нико се није
95 50 70 5	5	ако је праг 50 поена, пласирали су

Напомена: У бар 15 од 20 тест примера, бројеви m и n ће бити мањи од 200.

Решење: Погледати задатак за 6. разред

Окружно такмичење ученика основних школа из рачунарства - осми разред

1. [numeracija] Књига има n страна. Колико је цифара употребљено у њиховој нумерацији (она, наравно, креће од 1)? Са стандардног улаза се учитава број n (1 ≤ n ≤ $2 \cdot 10^8$). На стандардни излаз исписати тражени број цифара.

Примери

Улаз:	*Излаз:*	*Улаз:*	*Излаз:*
10	11	785	2247

Објашњење: Нумерација 10 страна се врши бројевима 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 и у њима укупно има 11 цифара.

Решење: Погледати задатак за 7. разред

2. [prag] Државна комисија треба да одреди праг за пролазак такмичара са окружног на државно такмичење. Пошто је информатика постала обавезан предмет у основним школама, број такмичара је јако велики. Администраторку Мају која одржава табелу са резултатима стално питају који би број такмичара прошао даље када би праг пролазности био толико и толико поена (даље се пласирају сви ученици чији је број поена већи или једнак прагу). Одлучила је да напише програм који даје одговор на та питања. Са стандардног улаза учитава се број такмичара n (1 ≤ n ≤ 50000), а затим и поени такмичара (природни бројеви), задати у сортираном редоследу од највећег до најмањег и раздвојени размацима. Након тога се учитава број m (1 ≤ m ≤ 50000) који представља број питања на која Маја треба да одговори, а затим и m бројева раздвојених размацима за које је потребно дати одговор колико би се такмичара пласирало када би се тај број узео за праг. На стандардни излаз исписати тражене бројеве такмичара који су се пласирали, у посебном реду за сваки праг.

Примери

Улаз:	*Излаз:*	*Објашњење:*
5	0	ако је праг 95 поена, нико се није
89 73 73 56 23	4	ако је праг 50 поена, пласирали су
4	3	ако је праг 95 поена, нико се није
95 50 70 5	5	ако је праг 50 поена, пласирали су

Напомена: У бар 15 од 20 тест примера, бројеви m и n ће бити мањи од 200.

Решење: Погледати задатак за 6. разред

3. [origami] Јована жели да слаже оригами. Има папир правоугаоног облика, чије су димензије страница цели бројеви. Пошто се сви модели које она уме да сложи слажу од папира квадратног облика она жели да исече што више квадрата, при чему јој је јако важно да ти квадрати буду што већи, да би их лакше пресавијала (није неопходно да сви изрезани квадрати буду исте димензије). Колико квадрата на тај начин може да добије? Са стандардног улаза се учитавају димензије

страница (два природна броја мања од $2 \cdot 10^9$, сваки у посебном реду). На стандардни излаз исписати тражени број квадрата (он ће сигурно бити мањи од $2 \cdot 10^9$).

Пример

Улаз: Излаз:

46 8

18

Објашњење: исецају се два квадрата димензије 18, затим један квадрат димензије 10, један квадрат димензије 8 и четири квадрата димензије 2.

Програмски језик C++

```
#include <iostream>

using namespace std;

int main() {
  int a, b;
  cin >> a >> b;
  int bk = 0;
  while (b != 0) {
    bk += a / b;
    int ost = a % b;
    a = b;
    b = ost;
  }
  cout << bk << endl;
}
```

Програмски језик Python

```
a = int(input())
b = int(input())
bk = 0
while b != 0:
    bk = bk + a // b
    ost = a % b
    a = b
    b = ost
print(bk)
```

4. [koferi] На траци на аеродрому се налазе кофери путника, сложени један до другог. Радници желе да утоваре неке кофере са траке на њихово возило и да их превезу до авиона и бирају кофер од којег започињу утовар. Када крену да товаре кофере, они товаре редом све узастопне кофере са траке (ни један кофер не смеју да прескоче), све док не попуне возило. Напиши програм који одређује све могућности да се кофери утоваре тако да се возило искористи што боље тј. да укупна тежина утоварених кофера буде једнака носивости возила.

У првој линији стандардног улаза налази се природни број z (такав да је $1 \leq z \leq 10^6$) који представља носивост возила. У другој се налази број кофера n ($2 \leq n \leq 5 \cdot 10^5$), а у трећој масе кофера (позитивни природни бројеви мањи од 100), раздвојени размаком. Исписати све редне бројеве кофера од којих могу да започну утовар тако да возило буде потпуно попуњено (кофери на траци се броје од нуле), поређане растуће. Водити рачуна о ефикасности решења.

Пример

Улаз:	*Излаз:*	*Објашњење:*
125	2	Ако крену од кофера број 2, спаковаће
10	4	Ако крену од кофера број 4, спаковаће
35 40 25 50 50 50 25 35 15 35	5	Ако крену од кофера број 5, спаковаће 50+25+35+15

Програмски језик C++

```
#include <iostream>
#include <vector>

using namespace std;

int main() {
  // ubrzavamo ucitavanje
  ios_base::sync_with_stdio(false);


  // ucitavamo trazeni zbir
  int trazeniZbir;
  cin >> trazeniZbir;


  // izracunavamo parcijalne sume elemenata niza
  int n;
  cin >> n;
  vector<int> S(n+1);
  for (int i = 0; i < n; i++) {
    int x;
    cin >> x;
    S[i+1] = S[i] + x;
  }


  // u sortiranom nizu parcijalnih suma trazimo da li postoje dva
  // elementa cija je razlika jednaka trazenom zbiru
  int l = 0, d = 1;
  while (d <= n) {
    if (S[d] - S[l] < trazeniZbir) {
      d++;
    } else if (S[d] - S[l] > trazeniZbir) {
      l++;
    } else {
      cout << l << endl;
      l++;
    }
  }
  return 0;
}
```

```
#include <iostream>

using namespace std;

int main() {
  // ubrzavamo ucitavanje
  ios_base::sync_with_stdio(false);


  // ucitavamo trazeni zbir
  int trazeniZbir;


  cin >> trazeniZbir;


  // ucitavamo elemente niza
  int n;
  cin >> n;
  int a[50000];
  for (int i = 0; i < n; i++)
    cin >> a[i];


  // granice segmenta
  int i = 0, j = 0;
  // zbir segmenta
  int zbir = a[0];
  while (true) {
    // na ovom mestu vazi da je zbir = sum(ai, ..., aj) i da
    // za svako i <= j' < j vazi da je sum(ai, ..., aj') < trazeniZbir


    if (zbir < trazeniZbir) {
      // prelazimo na interval [i, j+1]
      j++;
      // ako takav interval ne postoji, zavrsili smo pretragu
      if (j >= n)
        break;
      // izracunavamo zbir intervala [i, j+1] na osnovu zbira intervala [i, j]
      zbir += a[j];
    } else {
      // ako je zbir jednak trazenom, vazi da je sum(ai, ..., aj) =
      trazeniZbir
      // pa prijavljujemo interval
      if (zbir == trazeniZbir)
            cout << i << endl;
      // prelazimo na interval [i+1, j]
      // izracunavamo zbir intervala [i+1, j] na osnovu zbira intervala [i, j]
        zbir -= a[i];
        i++;
    }
  }


  return 0;
}
```

Програмски језик Python

```
trazeniZbir = int(input())
n = int(input())
a = list(map(int, input().split()))
i = 0
j = 0
zbir = a[0]
while True:
     if zbir < trazeniZbir:
      j += 1
      if j >= n:
        break;
      zbir += a[j]
     else:
       if zbir == trazeniZbir:
           print(i)
       zbir -= a[i]
       i += 1
```

ДРУШТВО
МАТЕМАТИЧАРА СРБИЈЕ

ПРОГРАМ ЗА ТАКМИЧЕЊЕ ИЗ ПРОГРАМИРАЊА ЗА УЧЕНИКЕ ОСНОВНИХ ШКОЛА

Циљ овог програма је да се:

1) одреди оквир у коме ће се састављати и бирати такмичарски задаци;

2) такмичарима и онима који планирају и реализују припреме за такмичење предочи у ком оквиру могу да се очекују задаци на такмичењима.

Циљ овог програма није да сам по себи представља програм припрема за такмичење, већ само да одреди елементе које би свакако требало обухватити у припреми за такмичење. Познавање наведених тема може да помогне у фокусању припрема на одређена подручја и знања.

На такмичењу, учесници се срећу са задацима који траже промишљање о датом проблему. Такође такмичари се срећу и са начинима како осмишљено решење записати у одабраном програмском језику.

Сви задаци који се задају на такмичењима су направљени да буду алгоритамски по природи и да се решавају у форми конзолних апликација. Формат улазних и излазних података су прецизно задати. Улазни подаци се учитавају са стандардног улаза, и они ће у тест примерима за евалуацију бити у описаном облику. Коректност улазних података није потребно проверавати у програму. Такмичарев програм мора исписивати излазне податке искључиво у траженом формату на стандардни излаз.

По питању елемената програмског језика овде описана очекивана знања и вештине су довољни за решавање свих задатака, што не искључује да одређена додатна знања и вештине у вези елемената програмског језика могу бити корисна за такмичара.

По питању техника програмирања и алгоритама се оставља могућност да тежи задаци буду тешки већини такмичара, да би се постигло адекватно рангирање између најбољих такмичара, а то се посебно односи на више нивое такмичења.

1.1 Очекивано знање према нивоима такмичења

У табели 1 је приказан очекивани ниво знања према нивоима такмичења.

Табела 1. Очекивано знање према нивоима такмичења

● – може се очекивати
◑ – може се очекивати у најтежем задатку

	Ниво такмичења						
	I катег.- општин.	I катег.- окружно	I катег.- државно	II катег.- општин.	II катег.- окружно	II катег.- државно	СИО
Петље	◑	●	●	●	●	●	●
Низови		◑	●	●	●	●	●
Бројеви у покретном зарезу				●	●	●	●
Матрице			◑	◑	●	●	●
Напредне технике програмирања I			◑		◑	●	●
Напредне технике програмирања II						◑	●

1.2 Променљиве и изрази за основне типове података

Такмичар зна да користи променљиве, доделу вредности и изразе у којима се појављују целебројне вредности, вредности у покретном зарезу, карактери, стрингови и истинитосне вредности.

Такмичар зна да користи целобројне типове чији опсег одговара 32-битном означеном целом броју и разуме разлику и рад са ширим и ужим регистром.

Такмичар зна да користи барем један тип за бројеве у покретном зарезу који може бити једноструке или двоструке прецизности.

Такмичар зна како се у програмском језику изводе следеће операције:

- сабирање, множење и одузимање целих бројева и бројева у покретном зарезу;
- унарни минус, апсолутна вредност и знак целог броја и броја у покретном зарезу;
- целобројно дељење и остатак при дељењу целих бројева;
- дељење бројева у покретном зарезу и остатак при дељењу бројева у покретном зарезу;
- квадратни корен броја у покретном зарезу;
- заокруживање броја у покретном зарезу на цео број и то: наниже, навише и на ближи („floor“, „ceiling“ и „round“);
- конверзије између бројевних типова;
- конверзија стринга у број и броја у стринг коришћењем декадног бројног записа;
- дужина стринга;
- надовезивање стрингова и издвајање подстринга;
- конверзија из карактера у његов целобројни кôд и обрнуто;
- поређење вредности (мање/веће/једнако) за сваки од типова;
- логичке операције дисјункција („или“), конјункција („и“) и негација.

Уколико изабрани програмски језик нема директну подршку за неке од наведених операција, такмичар треба да зна како одговарајућу обраду да изведе користећи расположиве могућности програмског језика.

Такмичар разуме да операције са бројевима у покретном зарезу најчешће дају приближан резултат, а посебно зна како да правилно пореди бројеве у покретном зарезу и како да конвертује број у покретном зарезу у цео број.

1.3 Улаз и излаз

Такмичар зна да са стандардног улаза учитава целобројне вредности, вредности у покретном зарезу и стринговске вредности када је свака појединачна вредност записана у посебном реду или када су вредности раздвојене сепаратором (белина, зарез, двотачка,...).

Такмичар зна да на стандардни излаз исписује бројеве, стрингове и карактере и да их при томе распоређује у редове.

Приликом исписивања вредности у покретном зарезу такмичар зна да постави формат исписивања тако да се користи одређен број цифара иза децималне тачке.

1.4 Гранања

Такмичар зна да користи контролну структуру гранања „if“ са и без „else“. Такмичар зна да користи гранање у оквиру израза и вишеструко гранање уколико изабрани програмски језик то подржава.

Такмичар зна да имплементира сложене услове који захтевају угњеждена гранања и/или сложене логичке изразе у условима и/или увођење додатних истинитосних промељивих.

1.5 Петље

Такмичар зна да користи „for“ и „while“ петље.

Такмичар зна да у решењу примени петљу са вишеструким условима за излазак, односно останак у петљи, као и да правилно одреди шта треба да се уради пре уласка у петљу, шта у телу петље, а шта након изласка из петље.

Такмичар зна да у решењу примени итеративне поступке који захтевају вишеструке петље укључујући и комбинацију са вишеструким гранањима.

1.6 Функције, процедуре, методе и потпрограми

Такмичар зна како да део кода издвоји у функцију, процедуру, метод, односно потпрограм (термини зависе од програмског језика) како би смањио понављање кода и програм учинио прегледнијим.

1.7 Низови

Такмичар зна да користи низовске типове података.

Такмичар зна да секвенцу вредности истог типа учита у низ, односно испише на основу вредности елемената низа.

Такмичар зна да секвенцијално обради елементе низа и да при томе према потреби мења вредности елемената тог низа, поставља елементе другог низа, рачуна статистике попут минимума, максимума, збира, производа, просека и слично.

Такмичар да зна да преписује делове низа укључујући промену редоследа елемената.

Такмичар зна да проверава услове који су изражени као међусобне релације елемената низа (на пример, провера да ли је низ уређен, симетричан,...).

1.8 Матрице

Такмичар зна да користи матрице (дводимензионе низове) било као низове низова или као посебан тип уколико програмски језик то потржава.

Такмичар зна да секвенцу вредности истог типа учита у елементе матрице по врстама или по колонама, односно да испише на основу вредности елемената матрице.

Такмичар зна да секвенцијално обрађује елементе матрице (по врстама, по колонама и сл.) и да при томе према потреби мења вредности елемената матрице, поставља елементе друге матрице, рачуна статистике (попут минимума, максимума, збира, производа, просека и слично) по врстама или по колонама и резултате смешта у низ или их непосредно даље обрађујуе.

Такмичар зна да преписује правоугаоне делове матрице укључујући промену редоследа елемената.

Такмичар зна да проверава услове који су изражени као међусобне релације елемената матрице (на пример да ли је матрица симетрична).

1.9 Основне технике програмирања

Такмичар зна да решава задатке који се заснивају на опису реалног или замишљеног догађаја, разумевању описаних односа и примени елементарних физичких и математичких законитости.

Такмичар зна да решава задатке у којима је потребно осмишљавање алгоритма примерене сложености.

Такмичар влада техником коришћења целобројног дељења, односно дељења са заокруживањем и остатка при дељењу у решавању проблема који су засновани на неопходности заокруживања бројева.

Такмичар зна да у решењу реализује релације и операције из теорије бројева као што су одређивање простих бројева, растављање на просте бројеве, највећи заједнички делилац и слично.

Такмичар влада техником растављања броја на цифре декадног записа и израчунавања броја када се могу одредити цифре декадног записа.

Такмичар влада техником формирања римског записа броја и израчунавања броја када се могу одредити цифре римског записа.

Такмичар разуме шта значи када су одређени објекти распоређени у врсте и колоне или када је правоугаоник подељен на једнаке квадрате (поља) попут шаховске табле и у тако формулисаним задацима:

- зна да користи редне бројеве врста и колона у одређивању релација између положаја објеката укључујући аналогију кретања шаховских фигура;
- разуме дефинисање правоугаоног подручја (скупа поља, односно објеката) и подручја другог карактеристичног облика и зна да рашава проблеме засноване на својствима и

односима таквих подручја укључујући и одређивање броја поља у троугаоним формама (где се користи рачунање збира узастопних бројева);

- зна да у решењу реализује симулацију кретања на основу секвенце вредности које описују потезе према задатим правилима.

Такмичар зна да у решењу реализује сортирање низа алгоритмом који има квадратну сложеност или мању, као и да користи сортирање у решавању проблема.

Такмичар зна да у решењу реализује линеарно претраживање низа или матрице.

Такмичар зна да у решењу реализује рачунање са временима и датумима.

Такмичар зна да решава задатке у којима се користе правила првенста пролаза у раскрсници, уколико су та правила објашњена у задатку.

Такмичар зна да решава задатке у којима је потребно одредити односе тачака, дужи и правоугаоника коме су ивице паралелне осама координатног система у равни.

1.10 Напредне технике програмирања I

Такмичар зна да оцени сложеност алгоритма који примењује, како по питању времена извршавања, тако и по питању заузимања меморије.

Такмичар детаљно познаје опсег вредности, прецизност, заузеће меморије и перформансе основних операција за основне типове података које нуди програмски језик и на основу тога уме да изабере одговарајући тип.

Такмичар зна да у решењу реализује ефикасне алгоритме сортирања и претраге.

Такмичар влада техником рекурзивног свођења на потпроблеме.

Такмичар влада техником рекурзивне претраге укључујући претрагу по дубини и по ширини, као и технику претраге са враћањем (backtracking).

Такмичар зна да у решењу реализује алгоритме из области комбинаторике.

Такмичар зна да решава проблеме у којима је потребно одредити односе тачака, дужи, троуглова и правоугаоника у равни.

Такмичар зна да решава проблеме у којима се појављује структура графа у улазним подацима уколико се алгоритми обраде графова своде на непосредну примену претходно наведених техника и алгоритама.

Такмичар зна да у решењу реализује алгоритам бинарне претраге.

1.11 Напредне технике програмирања II

Такмичар влада техником динамичког програмирања и техником мемоизације рекурзивних функција.

Такмичар зна да у решењу реализује специјализоване алгоритме за обраду графовова.

Такмичар зна да у решењу реализује геометријске алгоритме који заснивају на односима дужи, тачака, полигона и полиедара.

Такмичар зна да у решењу реализује рачунање са великим бројевима.

Такмичар зна да решава проблеме у којима треба применити идеје из познатих напредних алгоритама сортирања и претраге.