Општинско такмичење ученика основних школа из рачунарства - пети разред (22.02.2020.)

1. [**РОЂЕНДАН**] Јелена се припрема за такмичење из историје. Ако је њен омиљени историјски херој прославио R-ти рођендан G-те године, напиши програм који одређује које године је тај херој рођен. Са улаза се учитавају природни бројеви R, G (сваки у посебном реду, 1≤R≤98, 100≤G≤2020)

Улаз:	*Излаз:*	*Улаз:*	*Излаз*	*Улаз:*	*Излаз*	*Улаз:*	*Излаз:*	*Улаз:*	*Излаз:*
10	2010	40	1841	42	1978	80	1892	50	1967
2020		1881		2020		1972		2017	

Решење:
Ако је, на пример, Лаза Костић, прославио G-те године (G=1881) R-ти рођендан, то значи да је рођен R година раније. Година рођења се израчунава тако да од G-те године се одброји R година уназад, тј. од G се одузме R.

C++
```
#include <iostream>

using namespace std;

int main()
{ int R, G;
  cin >>R >> G;
  int rodjendan;
  rodjendan=G-R;
  cout << rodjendan << endl;
    return 0;
}
```

Python
```
R=int(input())
G=int(input())
rodjendan=G-R
print(rodjendan)
```

2. [**МНОЖЕЊЕ**] Мали Ика учи множење једноцифрених бројева. Али, Ика зна само бројеве прве десетице. Када измножи два броја, он тачно запише резултат само ако је производ у првој десетици. Иначе, напише 10. Напиши програм који за два унета једноцифрена броја исписује производ који ће записати мали Ика. У случају да тачан резултат не припада првој десетици, у новом реду исписати и за колико је тачан резултат већи од броја 10. Са улаза се учитавају једноцифрени бројеви m и n (сваки у посебном реду, 1<=m, n<11).

Улаз:	*Излаз:*	*Улаз:*	*Излаз*	*Улаз:*	*Излаз*	*Улаз:*	*Излаз:*	*Улаз:*	*Излаз:*
2	10	3	10	3	3	3	9	2	10
5		10	20	1		3		8	6

Решење:
Помножимо два унета броја. Ако је резултат множења у првој десетици, онда је потребно исписати тај резултат. Ако је резултат множења већи од 10 онда је потребно (у засебним редовима) исписати два броја на стандардни излаз: број 10 у првом реду, разлику производа и броја 10 у другом реду.

C++
```
#include <iostream>

using namespace std;

int main()
{
```

```
    int m,n;
    cin >>m>>n;
    int proizvod;
    proizvod=m*n;

    if (proizvod > 10)
       cout <<"10"<<endl<<proizvod-10<<endl;
    else cout <<proizvod <<endl;

    return 0;
}
```

Python

```
m=int(input())
n=int(input())
proizvod=m*n
if proizvod > 10:
    print(10)
    print(proizvod-10)
else:
    print(proizvod)
```

3. [**ЦЕНТУМ**] Век је временско раздобље од 100 година. У овом задатку претпоставите да у нашој ери (н.е.) први век је трајао од 1. јануара 1. године до 31. децембра 100.године, као и да други век н.е. је почео 1.јануара 101. године. Завршетак другог века н.е. је 31.децембра 200. године. Ми смо тренутно у 21.веку који је почео 1.јануара 2001. године. Напишите програм који за дату годину (1<=G<=2305) исписује ком веку припада та година.

Улаз:	*Излаз:*	*Улаз:*	*Излаз:*	*Улаз:*	*Излаз:*	*Улаз:*	*Излаз:*	*Улаз:*	*Излаз*
313	4	2020	21	1499	15	1500	15	1601	17

Решење:

Ознака века се поклапа са ознаком стотине у којој се налази број који означава текућу годину. Зато је довољно гледати целобројни количник при дељењу са 100. Када је година дељива са 100, онда се век добија на основу тог количника. Ако година није дељива са 100, онда се добијени количник увећа за 1, јер је наша ера почела првим, а не нултим веком.

C++

```
#include <iostream>
using namespace std;
int main()
{   int godina;
    cin >> godina;

    int vek;
    if (godina %100==0)
        vek=godina/100;
    else vek=godina/100+1;

    cout << vek << endl;
    return 0;
}
```

Python

```
godina=int(input())
if godina %100==0:
```

```
    vek=godina//100
else:
    vek=godina//100+1;

print(vek)
```

4. **[ПРЕКИДАЧ]** Прекидач укључује сијалицу код степеништа када се на њега притисне, а сијалица се аутоматски искључује 60 секунди након последњег притиска на прекидач. Колико секунди је светлела сијалица ако је прекидач притиснут у тренуцима T_1, T_2 и T_3? Са улаза се учитавају цели позитивни бројеви T_1, T_2 и T_3 (сваки у посебном реду, $1 <= T_1, T_2, T_3 <= 1\,000\,000$) времена притисака на прекидач изражена у секундама протеклим од неког тренутка у прошлости. На стандардни излаз испишите један цео број, укупно време светљења сијалице изражено у секундама.

Улаз:	*Излаз:*	*Улаз:*	*Излаз:*	*Улаз:*	*Излаз:*
158 300 121	157	11 25 48	97	3 100 182	180

Решење:
Задатак је лакше решити ако преуредимо временске тренутке T_1, T_2 и T_3 тако да важи $T_1 \leq T_2 \leq T_3$. Укупно време светљења можемо да разложимо на три дела. У сваком делу светљење почиње притиском на прекидач, а завршава се или следећим притиском на прекидач или истеком 60 секунди, у зависности од тога шта се пре догоди. Тако је трајање првог дела једнако мањем од бројева ($T_2 - T_1$) и 60, трајање другог дела мањем од бројева ($T_3 - T_2$) и 60, а трајање трећег дела је увек 60.

C++

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

int main() {
    int t1, t2, t3;
    cin >> t1 >> t2 >> t3;

    if (t1 > t2) swap(t1, t2);
    if (t2 > t3) swap(t2, t3);
    if (t1 > t2) swap(t1, t2);

    cout << (min(60, t2-t1) + min(60, t3-t2) + 60) << endl;
    return 0;
}
```

Python

```
t1, t2, t3 = sorted([int(input()) for i in range(3)])
print(min(60, t2-t1) + min(60, t3-t2) + 60)
```

Општинско такмичење ученика основних школа из рачунарства - шести разред (22.02.2020.)

1. [**САБИРАЊЕ**] Мали Ика је научио бројеве прве стотине. Сваког дана Ика броји од 1 до 100 и то по неколико пута узастопно. Напишите програм који ће на основу претпоследњег и последњег броја који је Ика изговорио исписати следећи број који ће Ика изговорити током бројања. Са улаза се учитавају два броја m и n (сваки у посебном реду, 1<=m, n<101).

Улаз:	*Излаз:*	*Улаз:*	*Излаз*	*Улаз:*	*Излаз*
3	5	2	4	98	100
4		3		99	

Решење:
Након уноса оба броја, довољно је увећати други број за 1 (осим ако тај други број није једнак 100) и исписати на стандардни излаз. Зато најпре проверимо да ли је други број једнак 100. У том случају, потребно је исписати 1, зато што ново одбројавање бројева прве стотине креће од броја 1 (јер је тако описан поступак узастопног одбројавања).

C++
```
#include <iostream>
using namespace std;

int main() {
    int m,n;
    cin >> m >> n;
    if (n==100) cout <<"1"<<endl;
    else cout <<n+1<<endl;
    return 0;
}
```

Python
```
m=int(input())
n=int(input())
if n==100:
    print(1)
else:
    print(n+1)
```

2. [**МЕД**] Пчелар Дуле је најпознатији произвођач меда. Да би мед био укусан, Дуле нам је открио тајну. Мед се чува у три посебне тегле. У прву теглу може се улити највише A килограма меда, у другу B килограма, у трећу C килограма. Али, мед се обавезно улива у тегле редом почевши од прве тегле, не прелазећи на следећу теглу док се претходна не напуни. Ако Дуле има N килограма меда које улива у тегле, одреди колико је у којој тегли било меда након што је уливено свих N килограма. У првом реду налази се природан број A ($1 \le A \le 10$), количина меда која стане у прву теглу. У другом реду налази се природан број B ($1 \le B \le 10$), количина меда која стане у другу теглу. У трећем реду налази се природан број C ($1 \le C \le 10$), количина меда која стане у трећу теглу. У четвртом реду налази се природан број N ($1 \le N \le A+B+C$), количина меда за уливање. На стандардном излазу написати три природна броја један испод другог. Први број је количина меда у првој тегли, други број је количина у другој, трећи број је количина меда у трећој тегли.

Улаз:	Излаз:	Улаз:	Излаз:	Улаз:	Излаз:	Улаз:	Излаз:
1	1	3	3	3	3	2	2
1	1	7	7	7	7	8	4
1	1	8	0	5	1	6	0
3		10		11		6	

Решење:
Замислимо да су испред нас три празне тегле (`teglaA=teglaB=teglaC=0`). Знамо да у прву теглу можемо улити највише A килограма меда. Ако је почетна маса меда N таква да се може напунити до врха прва тегла, онда ћемо у прву

теглу улити A килограма меда. Преостала маса меда за уливање је `N=N-A`. Горе описани поступак применимо над другом теглом,а потом над трећом теглом. Због услова $1 \leq N \leq A+B+C$ знамо да неће остати мед који није уливен у тегле.

C++

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

int main() {
    int a, b, c, n;
    cin >> a >> b >> c >> n;
    int teglaA = 0, teglaB = 0, teglaC = 0;
    if (n>=a)
        teglaA = a;
    else
        teglaA = n;
    n = n - teglaA;

    if (n>=b)
        teglaB = b;
    else
        teglaB = n;
    n = n - teglaB;

    teglaC = n;

    cout << teglaA << endl << teglaB << endl << teglaC << endl;
      return 0;
}
```

Python

```
A=int(input())
B=int(input())
C=int(input())
N=int(input())
teglaA=teglaB=teglaC=0

if N>=A:
    teglaA=A
    N=N-A
else:
    teglaA=N
    N=0

if N>=B:
    teglaB=B
    N=N-B
else:
    teglaB=N
    N=0

teglaC=N


print(teglaA)
print(teglaB)
print(teglaC)
```

3. **[СЕСТРЕ]** Ана, Бранка, Весна и Гоца су четири сестре. Ана је виша од Бранке исто колико Бранка од Весне, као и Весна од Гоце. Одредити висину Бранке и Весне ако је позната висина Ане и Гоце. У првом реду стандардног улаза је цео број ***A***, Анина висина у центиметрима (125≤ A≤190). У другом реду стандардног улаза је цео број ***G***, Гоцина висина у центиметрима (85≤G≤160). Вредности ***A*** и ***G*** су такве да су висине осталих сестара такође целобројне. У првом реду стандарног излаза исписати висину Бранке, а у другом висину Весне.

Улаз:	*Излаз:*
187	177
157	167

Решење:

Означимо разлику у висини Ане и Бранке са ***R***. Тада је ***A = G + 3R***, односно ***R = (A - G) / 3***. Након што смо одредили вредност ***R***, висина Бранке се једноставно одређује као ***B = A - R***, а слично томе, Веснина висина је ***V = B - R***.

C++

```
#include <iostream>
using namespace std;
int main() {
   int a, g;
   cin >> a >> g;
   int razlika = (a-g) / 3;
   int b = a - razlika;
   int v = b - razlika;
   cout << b << endl;
   cout << v << endl;
   return 0;
}
```

Python

```
a = int(input())
g = int(input())
razlika = (a-g) // 3
b = a - razlika
v = b - razlika
print(b)
print(v)
```

4. [**ЦВРКА**] Кукавица Цврка из зидног сата се оглашава на пун сат онолико пута колико има сати, а на пун сат и 15, 30 или 45 минута по једном. Бака Ката је навила сат у ***S*** сати и ***M*** минута. Када ће се Цврка следећи пут огласити, ако знамо да се она никад не оглашава чим је навијена? У првом реду стандардног улаза дат је цео број ***S*** (1 <= S <= 10). У другом реду стандардног улаза дат је цео број ***M*** (0 <= M <= 59). Исписати у једном реду стандардног излаза два цела броја, сат и минут следећег Цвркиног оглашавања, раздвојене једним размаком.

Улаз:	*Излаз:*	*Улаз:*	*Излаз:*
5	5 45	9	10 0
30		49	

Решење:

Означимо са `t` време у минутима протекло од последње поноћи. Број `t` можемо израчунати као `t = s*60 + m`. Време следећег Цвркиног оглашавања (у минутима протеклим од последње поноћи) је најмањи број који је већи од `t` и дељив са 15. Ово време можемо да израчунамо као `(t / 15 + 1) * 15`. Приметимо да је дељење овде било целобројно (оператор `//` у Пајтону), што значи да евентуални остатак намерно губимо). Остаје још да израчунато време претворимо у сате и минуте и испишемо.

C++

```
#include <iostream>
using namespace std;
int main() {
   int s, m;
   cin >> s >> m;
   int t = s*60 + m;
   t = (t / 15 + 1) * 15;
   int s2 = t / 60;
   int m2 = t % 60;
   cout << s2 << " " << m2 << endl;
   return 0;
}
```

Python

```
s = int(input())
m = int(input())
t = s*60 + m
t = (t // 15 + 1) * 15
s2 = t // 60
m2 = t % 60
print(s2, m2)
```

Општинско такмичење ученика основних школа из рачунарства - седми разред (22.02.2020.)

1. **[ОДСУТНА]** Ана, Бранка, Весна и Гоца су четири сестре. Ана је виша од Бранке исто колико Бранка од Весне, као и Весна од Гоце. Ана и још две од сестара су биле на систематском прегледу, а четврта није. Одредити висину одсутне сестре ако су познате висине остале три. У сваком од три реда стандардног улаза дат је по један цео број x (85 <= x <= 200), висине тријe сестара у центиметрима, међу којима је и Анина. Подаци су дати у произвољном редоследу, али тако да увек постоји јединствено целобројно решење. Исписати на стандардни излаз један цео број, висину четврте сестре у центиметрима.

Улаз:	*Излаз:*
176 146 *156*	`166`

Решење:

Задатак се знатно лакше решава ако прво уредимо три дате висине `a`, `b`, `c`, од највеће до најмање. Након уређивања, разлику првог и другог броја означимо са r_1, а разлику другог и трећег са r_2. Могућа су три случаја. У случају да недостаје Гоцина висина, вредности r_1 и r_2 ће бити једнаке, а Гоцина висина је $c-r_1$; ако недостаје Веснина висина, r_2 ће бити двоструко веће од r_1, а Веснину висину можемо израчунати као $b-r_1$ или $c+r_1$; у преосталом случају недостаје Бранкина висина и тада је r_1 двоструко веће од r_2, а Бранкина висина се израчунава као `a-r2`. Који је случај наступио, лако можемо да установимо поредећи вредности r_1 и r_2.

C++

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

int main() {
    int a, b, c;
    cin >> a >> b >> c;

    if (a < b) swap(a, b);
    if (b < c) swap(b, c);
    if (a < b) swap(a, b);

    int r1 = a - b;
    int r2 = b - c;

    if (r1 == r2) cout << c-r1 << endl;
    else if (r1 == 2 * r2) cout << a-r2 << endl;
    else cout << b-r1 << endl;

    return 0;
}
```

Python

```
c, b, a = sorted([int(input()) for i in range(3)])
r1 = a - b
r2 = b - c
if r1 == r2:
    print(c-r1)
elif r1 == 2 * r2:
    print(a-r2)
else:
    print(b-r1)
```

2. [**ЦАРЕ ГОСПОДАРЕ**] Некада давно Ваши родитељи су играли игру *Царе, царе, Господаре, колико је сати?* У тој игри Цар свој одговор даје у облику: ***M*** минута до ***H*** сати. Напишите програм који за унете бројеве ***M*** и ***H*** исписује тачно време у уобичајеном облику час:минут где час је двоцифрена ознака за број часова у 24-часовном формату, а минут је двоцифрена ознака за број минута у часу. Претпоставити да број M је такав да тражено време је увек у истом дану. Претпоставите да један дан траје од 00:00 до 23:59. Са улаза се учитавају природни бројеви ***M***, ***H*** (сваки у посебном реду, 0<=M<=1440, 0<=H<=23).

Улаз:	*Излаз:*	*Улаз:*	*Излаз:*	*Улаз:*	*Излаз*
23	22:37	15	09:45	55	21:05
23		10		22	

Решење:

Цар свој одговор даје у облику: ***M*** минута до ***H*** сати. Најпре ћемо тај одговор претворити у укупан број минута који је протекао од поноћи (`ukupnominuta=H*60-M`). Укупан број минута ћемо претворити у текући час и минут користећи чињеницу да један час има 60 минута (`cas=ukupnominuta/60;minut=ukupnominuta%60;`). На крају је потребно водити рачуна о форматираном испису два двоцифрена броја. Можемо користити посебне наредбе намењене испису целог броја у пољу минималне ширине 2 (нпр. у Пајтону print('(:02d):(:02d)'.format(cas,minut))) или уз помоћ наредби гранања дописивати водећу нулу у случају да вредности часа или минута јесу једноцифрени бројеви.

C++

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

int main()
{
    int M, H;
    cin >>M>>H;
    int ukupnominuta=H*60-M;
    int cas, minut;
    cas=ukupnominuta/60;
    minut=ukupnominuta%60;
    if (cas <10) cout <<"0";
    cout <<cas <<":";
    if (minut <10) cout <<"0";
    cout << minut << endl;
    return 0;
}
```

Python

```
M=int(input())
H=int(input())
ukupnominuta=H*60-M
cas=ukupnominuta//60
minut=ukupnominuta%60
if cas <10 and minut <10:
    print('0', cas, ':0', minut, sep='')
elif minut<10:
    print(cas,':0',minut,sep='')
elif cas <10:
    print('0',cas,':',minut,sep='')
else:
    print(cas, minut, sep=':')
```

3. [**КУПЦИ**] Менаџер једног великог ланца продавница жели да зна структуру прихода. Он је зато све куповине разврстао у мале, средње и велике на следећи начин: ако је износ рачуна за неку куповину ***X***, та куповина се сматра за малу ако је X <= M, за средњу ако је M < X <= V, а за велику ако је V < X. Менаџер жели да за посматрани период зна колики је укупан приход продавница од малих, средњих и великих куповина редом. Са стандардног улаза се у првом реду учитавају бројеви ***M*** и ***V*** (1 <= M < V <= 100000). У другом реду је цео број ***n***, укупан број куповина у свим продавницама у посматраном периоду (1 <= n <= 50). У трећем и последњем реду су износи рачуна за ***n*** куповина, цели позитивни бројеви до 200000 раздвојени по једним размаком. На стандардни излаз исписати три цела броја, сваки у посебном реду. Ови бројеви су укупни приходи од малих, средњих и великих куповина.

Улаз:	*Излаз:*
100 1000	188
5	512

88 1010 512 100 2500 3510

Решење:
Формирамо три суме (`sm, ss, sv`), које ће редом садржати приходе од малих, средњих и великих куповина. Након иницијализације све три суме, разматрамо једну по једну куповину, поредимо је са границама за мале, средње и велике куповине и разврставамо, а затим је додајемо на одговарајућу суму. На крају исписујемо све три суме.

C++
```
#include <iostream>
using namespace std;

int main() {
    int m, v, n, racun;
    cin >> m >> v >> n;
    int sm = 0, ss = 0, sv = 0;
    for (int i = 0; i < n; i++) {
        cin >> racun;
        if (racun <= m)
            sm += racun;
        else if (racun <= v)
            ss += racun;
        else
            sv += racun;
    }
    cout << sm << endl << ss << endl << sv << endl;
    return 0;
}
```

Python
```
s = input().split()
m, v = int(s[0]), int(s[1])
n = int(input())
racuni = list(map(int, input().split()))
sm, ss, sv = 0, 0, 0
for racun in racuni:
    if racun <= m:
        sm += racun
    elif racun <= v:
        ss += racun
    else:
        sv += racun
print(sm)
print(ss)
print(sv)
```

4. [**ГРИП**] Због сезонског грипа, мали Пера данас мери температуру тела сваких пола сата. Написати програм који ће за дато време првог мерења вредности температуре и N мерења, одредити и исписати највишу измерену температуру и тачно време када је та температура измерена. Можете претпоставити да су сва мерења извршена у истом дану, као и да ће постојати само једна највиша измерена температура. У првом реду стандардног улаза налази се један цео број H ($0 \leq H \leq 23$), час у коме је извршено прво мерење температуре. У другом реду налази се један цео број M ($0 \leq M \leq 59$), минут у ком је забележено прво мерење температуре. У трећем реду налази се један природан број N ($1 \leq N \leq 10$), број извршених мерења. У следећих N редова налази се по један природан број T ($1 \leq T \leq 42$), вредност измерене температуре у сваком следећем мерењу. У првом реду стандардног излаза исписати највишу измерену температуру. У другом реду испишите час, а у трећем реду минут у ком је измерена та највиша температура.
Улаз: Излаз: Улаз: Излаз: Улаз: Излаз: Улаз: Излаз:

14	39	14	39	0	39	10	37
0	15	0	14	0	0	21	11
3	0	3	30	2	30	3	21
37		36		38		35	
36		39		39		36	
39		37				37	

Решење

Након учитавања почетног времена и броја мерења `n`, тражимо највећи од наредних `n` бројева и његов редни број `i_max`. Након проласка кроз сва мерења, треба да одредимо време највише температуре, а оно се добија додавањем `i_max` пута по 30 минута на почетно време. У случају да је температура била највиша при првом мерењу, не треба додавати ништа на почетно време. Према томе, при одређивању редног броја највише температуре згодно је да се броји од 0 (редни број прве температуре је 0).

C++

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

int main() {
    int h, m, n, t;
    cin >> h >> m >> n;
    int tMax = 0, iMax = 0;
    for (int i = 0; i < n; i++) {
        cin >> t;
        if (t > tMax) {
            tMax = t;
            iMax = i;
        }
    }
    int ukMinuta = h*60 + m + 30*iMax;
    cout << tMax << endl << ukMinuta / 60 << endl << ukMinuta % 60 << endl;
    return 0;
}
```

Python

```
h = int(input())
m = int(input())
n = int(input())
t_max = 0
i_max = 0
for i in range(n):
    t = int(input())
    if t > t_max:
        t_max = t
        i_max = i

uk_minuta = h*60 + m + 30*i_max
print(t_max)
print(uk_minuta//60)
print(uk_minuta % 60)
```

Општинско такмичење ученика основних школа из рачунарства - осми разред (22.02.2020.)

1. **[ОДСУТНА]** Ана, Бранка, Весна и Гоца су четири сестре. Ана је виша од Бранке исто колико Бранка од Весне, као и Весна од Гоце. Ана и још две од сестара су биле на систематском прегледу, а четврта није. Одредити висину одсутне сестре ако су познате висине остале три. У сваком од три реда стандардног улаза дат је по један цео број x (85 <= x <= 200), висине трију сестара у центиметрима, међу којима је и Анина. Подаци су дати у произвољном редоследу, али тако да увек постоји јединствено целобројно решење. Исписати на стандардни излаз један цео број, висину четврте сестре у центиметрима.

Улаз:	*Излаз:*
176 146 156	166

Решење:
Видети решење првог задатка за седми разред.

2. [**ЧУДЕСНА ЦИФРА**] Кажемо да цифра неког броја је чудесна ако је њена вредност једнака позицији на којој се налази у броју. На пример, у броју 92 цифра два је чудесна, јер се налази на другој позицији. У броју 1838 постоје две чудесне цифре: 1 и 3 (цифра један је на првој позицији и цифра три је на трећој позицији). Написати програм који за задати природан број ($1 \le N \le 99999$) исписује колико је у њему чудесних цифри, а потом у следећем реду исписује и вредност тих цифри гледано слева надесно.

Улаз:	*Излаз:*	*Улаз:*	*Излаз:*	*Улаз:*	*Излаз:*
94	0	423	2 2 3	1838	2 1 3

Решење:
У овом задатку је згодније да се број N учита као стринг. Задатак решавамо у два пролаза кроз цифре броја N. У првом пролазу само бројимо чудесне цифре, па ако их има онда их у другом пролазу исписујемо. Услов да је `i`-та цифра слева чудесна је да бројчана вредност те цифре једнака `i` (односно једнака `i`+1 пошто цифре бројимо од 0).

C++

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

int main() {
    string a;
    cin >> a;
    int n = 0;
    for (int i = 0; i < a.size(); i++)
        if (a[i] - '0' == i+1)
            n++;

    cout << n << endl;
    if (n > 0) {
        for (int i = 0; i < a.size(); i++)
            if (a[i] - '0' == i+1)
                cout << a[i] << " ";

        cout << endl;
    }
      return 0;
}
```

Python

```
a = input()
n = 0
for i in range(len(a)):
```

```
    if int(a[i]) == i+1:
        n += 1
print(n)
if n > 0:
    for i in range(len(a)):
        if int(a[i]) == i+1:
            print(a[i], end=' ')
    print()
```

3. [**CAT**] Кукавица из зидног сата се оглашава на пун сат онолико пута колико има сати (најмање 1 а највише 12), а на пун сат и 15, 30 или 45 минута по једном. Бака Ката је навила сат у S_1 сати и M_1 минута, а сада је S_2 сати и M_2 минута. Од навијања сата је прошло мање од 12 сати. Колико пута се кукавица огласила од навијања? У првом реду стандардног улаза је број S_1 ($1 <= S_1 <= 12$). У другом реду стандардног улаза је број M_1 ($1 <= M_1 <= 59$, $M_1 \neq 15$, $M_1 \neq 30$, $M_1 \neq 45$). У трећем реду стандардног улаза је број S_2 ($S_1 < S_2 <= 12$). У четвртом реду стандардног улаза број M_2 ($1 <= M_2 <= 59$, $M_2 \neq 15$, $M_2 \neq 30$, $M_2 \neq 45$). На стандардни излаз исписати један цео број, број оглашавања кукавице.

Улаз:	*Излаз:*	*Улаз:*	*Излаз:*
3 16 4 1	6	7 16 9 28	23

Појашњење првог тест примера: Кукавица се огласила једном у 3:30, затим једном у 3:45 и четири пута у 4:00.
Појашњење другог тест примера: 1 + 1 + 8 + 1 + 1 + 1 + 9 + 1 = 23

Решење:
Задатак се може решити и помоћу циклуса, а овде ћемо показати аритметичко решење. Нека је $t_1 = 60*s_1 + m_1$, $t_2 = 60*s_2 + m_2$. Одредимо прво број различитих тренутака M, у којима се кукавица оглашавала. Тих тренутака има исто колико и времена (изражених у минутима) дељивих са 15, која су између t_1 и t_2. , дакле $M = t_2/15 - t_1/15$. Од тих M тренутака, њих $s_2 - s_1$ је било на пун сат. То значи да се кукавица на непун сат оглашавала $M - (s_2 - s_1)$ пута, сваки пут по једном. На овај број оглашавања треба још додати и оглашавања на пуне сате. На сваки пун сат кукавица се оглашава онолико пута колико има сати, па је број оглашавања на пуне сате једнак збиру (природних) бројева већих од s_1 а мањих од s_2. Знајући да је збир свих бројева од 1 до K једнак $K*(K+1)/2$, тражени број оглашавања на пуне сате је $(s_2*(s_2+1) - s_1*(s_1+1)) / 2$. Сабирањем броја оглашавања на пуне и непуне сате добијамо тражени укупан број оглашавања.

C++

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

int main() {
    int s1, m1, s2, m2;
    cin >> s1 >> m1 >> s2 >> m2;

    int brPunihSati = s2 - s1;
    int br_15_30_45 = (s2*60 + m2) / 15 - (s1*60 + m1) / 15 - brPunihSati;
    int ukNaPuneSate = (s2*(s2+1) - s1*(s1+1)) / 2;

    cout << ukNaPuneSate + br_15_30_45 << endl;
      return 0;
}
```

Python

```
s1 = int(input())
m1 = int(input())
s2 = int(input())
m2 = int(input())

br_punih_sati = s2 - s1
```

```
br_15_30_45 = (s2*60 + m2) // 15 - (s1*60 + m1) // 15 - br_punih_sati
uk_na_pune_sate = (s2*(s2+1) - s1*(s1+1)) // 2
print(uk_na_pune_sate + br_15_30_45)
```

4. [**КУПЦИ**] Менаџер једног великог ланца продавница жели да зна структуру прихода. Он је зато све куповине разврстао у мале, средње и велике на следећи начин: ако је износ рачуна за неку куповину ***X***, та куповина се сматра за малу ако је X <= M, за средњу ако је M < X <= V, а за велику ако је V < X. Менаџер жели да за посматрани период зна колики је укупан приход продавница од малих, средњих и великих куповина редом. Са стандардног улаза се у првом реду учитавају бројеви ***M*** и ***V*** (1 <= M < V <= 100000). У другом реду је цео број ***n***, укупан број куповина у свим продавницама у посматраном периоду (1 <= n <= 50). У трећем и последњем реду су износи рачуна за ***n*** куповина, цели позитивни бројеви до 200000 раздвојени по једним размаком. Три цела броја, сваки у посебном реду. Ови бројеви су укупни приходи од малих, средњих и великих куповина.

Улаз:	*Излаз:*
100 1000 5 88 1010 512 100 2500	188 512 3510

Решење:
Видети решење трећег задатка за седми разред.