problem	kvadrat	svemir	jmbg

source file	`kvadrat.pas`
	`kvadrat.c`
	`kvadrat.cpp`
		`svemir.pas`
		`svemir.c`
		`svemir.cpp`
			`jmbg.pas`
			`jmbg.c`
			`jmbg.cpp`

input data	stdin	library	stdin
output data	stdout	library	stdout
time limit (Athlon 64 3000+)	1 sec		
memory limit (heap)	32 MB		
memory limit (stack)	8 MB		
points	**40**	**50**	**60**
	150		

Magic square consists of 3x3 smaller squares, each containing a positive integer less than or equal to 20,000.

It has a property that the **sums** of the numbers in each row, column and both diagonals are **all equal**. Dave had one beautiful magic square, but someone took a few (**at most 3**) numbers out of it.

Write a program that will, for a given magic square with some missing numbers, determine the numbers missing.

input data

Each of the three lines of input contains three numbers i.e. description of the magic square.

Squares with missing numbers are denoted with **number 0**.

output data

Three lines of output should contain the correctly filled magic square.

examples

input

```
4 9 2
3 0 7
8 1 6
```

output

```
4 9 2
3 5 7
8 1 6
```

input

```
0 12 12
16 10 0
8 8 14
```

output

```
6 12 12
16 10 4
8 8 14
```

input

```
495 468 0
0 522 414
441 0 549
```

output

```
495 468 603
630 522 414
441 576 549
```

svemir

Space shuttle has lost its black-box somewhere in the universe, and the crew is trying to find it as soon as possible.

Signal emitted from the box is not strong enough to determine the exact position of the box. However, when the spaceship moves, the instruments can determine if the signal from the box is now stronger or weaker than before, and hence the crew knows **if the spaceship is now closer or further from the box**.

The universe is a three-dimensional space consisted of NxNxN small squares. Each square is represented by three coordinates, all of them are positive integers less than or equal to N.

At the beginning, spaceship is located in the square (1,1,1), and the black-box is at an different unknown location.

Write a program that will find the black-box (i.e. by moving the spaceship to the exact position of the black-box) with **at most 200 calls** to function **Pomak**.

library

You are given a library **svemlib** that contains two functions:

DajN - this function returns the value of the number N, $2 \leq N \leq 1{,}000{,}000{,}000$.

```
function DajN : longint;
int DajN(void);
```

Pomak - this function should be called with three arguments – coordinates of the position where we want to move the spaceship. All of the arguments must be positive integers less than or equal to N.

```
procedure Pomak(x,y,z : longint) : longint;
int Pomak(int x,int y,int z);
```

Function Pomak moves the spaceship to the given location and returns:

-1 if the spaceship is now closer to the black-box then before the movement

0 if the spaceship is on the same distance from the black-box

1 if the spaceship is now further from the black-box

If, by calling this function, spaceship moves to the exact position of the black-box, library will **regularly terminate** the execution of your program.

Every citizen in one country has his own identification number consisting of 19 digits in the format:

DDMMYYYYAAAAAAAAAAC

where digits DD denote day, MM month, and YYYY year of the birth.

Year of birth is a positive integer between 0001 and 9999 inclusive. Year is a leap year if it is divisible by 4, but not by 100 or if it is divisible by 400.

Digits denoted by A can be arbitrary, and C is a control-digit calculated by the following algorithm:

1. Denote all the digits in identification number, except the last one, with Z1...Z18
2. S = (10*Z1+9*Z2+8*Z3+ ... +2*Z9+10*Z10+9*Z11+8*Z12+ ... +2*Z18) mod 19
3. if S ≤ 9 then C = S, otherwise C = 19-S

A few digits have been erased from an identification number and replaced with a character '**X**'.

Write a program that will calculate the total number of **different correct identification numbers** corresponding to the given pattern.

input data

First and only line of input contains the given pattern.

output data

First and only line of output should contain the total number of different correct identification numbers from the problem statement.

Note: the result will fit into the 64-bit signed integer type (int64 in Pascal, long long in C/C++).

examples

input	**input**	**input**
XX0220051234567890X	XXXX200577XXXXXXX7X	0XX52X0512X456X8903
output	**output**	**output**
28	3650000000	946