NEW BULGARIAN UNIVERSITY

Department *Informatics*

Fourth Interuniversity NBU Programming Contest

4 April, 2020

Problem A. Music Problem

You have a *la minor* chord given as a string: `abcdefga` (la, si, do, re, mi, fa, sol, la). Write a program for generating another minor chord for a given starting tone. A minor chord is a sequence of 7 tones with the following intervals between the tones: 1, ½, 1, 1, ½, 1. For example, between *la* and *si* there is one tone, between *si* and *do* – half a tone. The piano has a black button between the two white buttons (*la* and *si*), which is marked as a# (A/la sharp) or b♭ (B/si flat) and is at half a tone up from *la* and half a tone down from *si*.

Input

For each test there is the starting tone of the required minor chord.

Output

For each test, output on a new line the whole chord, using sharp signs (#, black piano buttons) for the half-tones.

Example

Input	Output
`e`	`ef#gabc`

Problem B. Periodic Sequences

The theory of discrete dynamic systems studies the following situation: There is a function

$f : R \rightarrow R$. The orbit of a given point x is the infinite sequence:

$$x, f(x), f(f(x)), f(f(f(x))), \ldots$$

Sometimes the orbit of a point x is the n-periodic sequence, i.e. a sequence of the kind:

$$a_1, a_2, \ldots, a_n, a_1, a_2, \ldots, a_n, \ldots$$

Such a point x is called an n-periodic point. Using a computer to study the point orbits, given a point's final part we need to determine whether it is periodic and to find its period n, if it is periodic. We assume that the given part of the orbit is a sufficiently long sequence and if the point is n-periodic, the sequence contains at least $2n$ members. Write a program, which is given a finite sequence of whole numbers and determines the period n, if the sequence is part of the orbit of a periodic point.

Input

The first number of each test is the length of the finite sequence, maximum 1000. Subsequent lines contain the members of the sequence – whole numbers, less than 100, separated by an interval or a newline. The input contains up to 10 tests and finishes with the digit 0 on the last line.

Output

For each test, output a whole number on a new line – the period n or 0, if the sequence is not part of the point's periodic orbit. Since there can be many possible periods (if n is a period, then $2n$, $3n$, etc. are also periods), you need to output the length of the smallest one.

Example

Input	Output
4 2 2 2 2 8 1 2 1 2 1 2 1 1 14 2 3 1 2 3 1 2 3 1 2 3 1 2 3 0	1 0 3

Problem C. Point Groups

You have *n* points in the plane. Determine the maximum number of groups in which the points can be divided, such that the distance between any two points from different groups is greater than a given number *d*. This task needs to be solved for different values of *d*. Write a program to solve this.

Input

Each test will contain: *n* – the number of points, *D* – the maximum distance between groups of points and *2n* whole numbers – the points' coordinates. The standard input will contain many tests.

Limits

2 <= n <= 1000, 1 <= D <= 1000

The points' coordinates are in the interval [0, 1000]

Output

For each test output a separate line containing *D* numbers – the number of groups for *d = 1, 2, … D*.

Example

Input	Output
2 1 0 0 10 10 4 2 0 0 1 1 2 2 3 3	2 4 1

Problem D. n-Numbers

An *n*-number is a natural number, which has *n* ones in its binary representation. Write a program which finds all *n*-numbers in a given closed interval.

Input

For each test, read from the standard input the numbers *n* and *m* – the number of intervals, in which you need to find *n*-numbers. They are followed by *2m* whole numbers – the lower and upper bound of the closed interval, in which the *n*-numbers should be found.

Limits

Input numbers are whole numbers in the interval $[1, 10^6]$

Output

For each test the standard output should contain a line of *m* numbers – the number of *n*-numbers found in the corresponding interval in order.

Example

Input	Output
1 2 1 5 8 16	2 3

Problem E. Long Sequence

There is a numeric sequence:

$f_1 = f_2 = 1$;

$f_n = (a\, f_{n-2} + b\, f_{n-1}) \bmod n$, for $n = 3, 4, \ldots, m$.

(`mod` means modulus – the remainder from an integer division).

Given the whole positive numbers *a* and *b*, find the count of different numbers in the sequence.

Input

Each test case contains the numbers *a*, *b* and *m*.

Limits

$1 <= a, b <= 100$

$3 <= m <= 10^8$

Output

For each test case output the resulting count on a separate line.

Example

Input	Output
1 2 5 2 8 100	3 48

Problem F. Tri-angles

If you take 3 random natural numbers, they can either be or cannot be the sides of a right triangle (e.g. 3, 4, 5). If they cannot, then they can either be sides of an acute triangle (e.g. 1, 1, 1) or a obtuse triangle (e.g. 5, 5, 9). There is one other possibility – these numbers cannot be the sides of a triangle (e.g. 1, 2, 4). Given a set of natural numbers, find the number of right, acute and obtuse triangles, whose side lengths are elements in the set.

Input

For each test case, the standard input will contain the number of elements in the set and then the actual values.

Limits

All input numbers are less than 1001.

Output

For each test, output on a separate line three numbers separated by space: the number of right, acute and obtuse triangles, whose side lengths are elements of the set.

Example

Input	Output
3 3 4 5 4 5 3 4 5	1 0 0 2 2 0

Explanation

For the second test case, triangles have lengths: 5, 3, 4 and 3, 4, 5 – right; 5, 3, 5 and 5, 4, 5 – acute.

Problem G. Packets

A communication channel transports data packets, which need to be processed by your program. Each packet contains code and numeric data. The code is a keyword, which determines what operation should be performed on the numeric data – a sequence of whole numbers.

The codes and operations are:

`min` – the minimum number;
`max` – the maximum number;
`sum` – the sum of the numbers;
`num` – the numbers count;

These 4 codes form another 8, by adding the suffix **p** or **n**. The suffix **p** in the code changes the operation to process only positive numbers in the numeric data, and suffix **n** – only the negative.

Input

Each separate line of the standard input contains one packet: code and data – a sequence of whole numbers, separated by spaces.

Limits

Numeric data are whole numbers in the interval [-100, 100]. The length of the sequence containing the numeric data in the packet is a number between 0 and 100.

Output

For each packet, output on a separate line the result of the operation. The result of the operation on an empty sequence is the character `*`.

Example

Input	Output
`sum 10 20`	`30`
`min 3 1 7 -2`	`-2`
`max -9 -2 0 -2`	`0`
`num 0 0 0 0 0 0 0 0`	`8`
`sump 10 20 -1`	`30`
`minp 3 1 7 -2`	`1`
`maxp -9 -2 0 -2`	`*`
`nump 0 1 0 0 0 0 0 0`	`1`
`sumn 10 20`	`*`
`minn 3 1 7 -2`	`-2`
`maxn -9 -2 0 -2`	`-2`
`numn 0 0 1 0 0 0 0 0`	`*`
`sum`	`*`
`min`	`*`
`max`	`*`
`num`	`*`

Problem H. String Intervals

You have two strings *a* and *b*, containing lowercase Latin letters. Find the number of strings *x* with a given length *n*, for which *a* < *x* < *b*.

Input

For each test case a line of the standard input will contain the two strings *a* and *b*, and the number *n*.

Limits

The length of the strings *a* and *b* is in the interval [1, 1000].

1 <= *n* <= 1000

a < *b*

Output

For each test case, output on a separate line the required number of strings by modulo 26 (remainder from an integer division).

Example

Input	Output
`abc abcd 4`	`3`
`ab az 3`	`0`
`a b 1`	`0`

Problem I. Interval of Numbers

You have two numbers *a* and *b*, which are lower and upper boundary of an interval of whole numbers. Write a program that shows whether the result of applying the bitwise operator *EXCLUSIVE OR* (XOR) on all whole numbers in the interval [*a*, *b*] is odd or even number.

Input

For each test case, a line of the standard input will contain the values of *a* and *b*, separated by space.

Limits

$1 <= a < b <= 10^5$

Output

For each test case, output a line containing `Even` if the result of applying XOR on all whole numbers in the interval [*a*, *b*] is even, or `Odd` otherwise.

Example

Input	Output
1 10	Odd
5 15	Even
40 43	Even
22 96	Odd
41 75	Even
12 37	Odd
10 82	Even
9 84	Even
1 23	Even
49 67	Even

Problem J. Guard

A guard must keep safe its garden with valuable plants, having the shape of a polygon with no two sides crossing or touching each other. The polygon is described by the coordinates of its vertices. The garden has a fence on the edges of the polygon. The guard must move on a straight line situated entirely in the polygon, including its edges, and must be able to see each point of the fence from each point of the line. Write a program to find the maximum length of such a line.

Input

The first line of the standard input will contain the number of test cases. Each test case begins with a line containing the count of N polygon vertices. Each subsequent N lines contains the coordinates of one of the vertices in one of the possible orders of traversal – two numbers, which can have a fractional value.

Limits

2 < N < 101

Output

For each test case, the program should output a separate line containing the resulting length, rounded up to the 5th decimal digit. If such a line does not exist or is a point, the program should display 0.00000.

Example

Input	Output
2 4 0.0 0.0 10.0 0.0 10.0 10.0 0.0 10.0 8 3 0 0 4 1 5 3 1 4 1 6 5 7 4 4 0	14.14214 0.00000