Problem A. Barcode (Division 1 Only!)

Input file:	barcode.in
Output file:	barcode.out
Time limit:	2 seconds (3 for Java)
Memory limit:	256 Mebibytes

Vasya works for RIBCA (Research Institute of BarCodes Analysis). He is developing a new noise-proof barcode analysis system. The only disadvantage of the system is that it is limited with barcodes that are created in the following way.

Consider a grid $w \times h$ of black and white cells. Initially, all cells are black. Select no more than n rectangles with sides aligned with the grid lines. After that, for each selected rectangle, take all cells inside and invert them (white turns into black and vice versa).

Now Vasya wants to know the number of different barcodes that his analysis system can handle. Write a program to calculate it.

Input

The input consists of one or more test cases. Each test case consists of a single line containing three integers w, h and n — the size of the grid and the maximal number of rectangles ($0 \leqslant n \leqslant 2$, $1 \leqslant w, h \leqslant 10$).

The input will be terminated with a test case with $w = h = n = 0$ which should not be processed.

The total number of test cases will not exceed 10.

Output

For each test case write a single line with a single integer — the number of different barcodes.

Example

barcode.in	barcode.out
2 2 0 2 2 1 2 2 2 0 0 0	1 10 16

Problem B. Buses

Input file:	buses.in
Output file:	buses.out
Time limit:	5 seconds (6 for Java)
Memory limit:	256 Mebibytes

Vasya works for RIBBeRy (Research Institute of Buses and Bus Routes). He is writing a program that plans new routes. The problem he is solving at the moment is the following: given a set of places where route endpoints might be placed, find the number of suitable pairs of endpoints. A suitable pair is a pair for which a suitable route exists; a suitable route is any shortest route with length d within a fixed range ($l_i \leqslant d \leqslant r_i$). As the routes are intended to be placed in a city, the distance between two points is the sum of differences of their coordinates.

Help Vasya to write a program that finds the described number of pairs for a given set of endpoints and a set of ranges.

Input

The input consists of one or more test cases. Each test case starts with a line containing three integers n, k and z — the number of endpoints, the number of ranges and the size of the city ($1 \leqslant n \leqslant 5\,000$, $1 \leqslant k \leqslant 100\,000$, $1 \leqslant z \leqslant 1\,000\,000$). The following n lines contain pairs of integers (x_i, y_i) — coordinates of endpoints ($0 \leqslant x_i, y_i \leqslant z$). The next k lines contain pairs of integers (l_i, r_i) — ranges of distances to be checked ($1 \leqslant l_i \leqslant r_i \leqslant 2 \cdot z$).

The input will be terminated with a test case with $n = k = z = 0$, which should not be processed.

The sum of n's in the whole input doesn't exceed 5 000. The sum of k's in the whole input doesn't exceed 100 000. The sum of z's in the whole input doesn't exceed 1 000 000.

Output

For each test case write k lines containing the number of pairs of points for each range. Output for each pair of consecutive test cases should be separated by an empty line.

Example

buses.in	buses.out
3 3 3 0 0 0 3 1 2 1 3 1 2 1 1 2 2 2 0 2 0 2 1 4 1 4 0 0 0	3 1 0 0 0

Problem C. Euclid (Division 1 Only!)

Input file:	`euclid.in`
Output file:	`euclid.out`
Time limit:	2 seconds (3 for Java)
Memory limit:	256 Mebibytes

Vasya works for RIA (Research Institute of Archaeology). During recent excavations around Alexandria, Vasya found an interesting paper. He suggested it to be a game log between two Euclid's scholars. Euclid selected some integer number X. Then the scholars started writing out *euclid-maximal* pairs of natural numbers with respect to X. The scholar who could not write out any new pair lost the game.

We call a pair (a_m, b_m) *euclid-maximal* with respect to X if $1 \leqslant a_m \leqslant b_m \leqslant X$ and Euclid's algorithm for finding the greatest common divisor makes maximal possible number of steps among all pairs of integers (a, b) such that $1 \leqslant a \leqslant b \leqslant X$. The number of steps is the final value of variable `step` in the following pseudocode:

```
function gcd (a, b):
|   step := 0
|   while a > 0 and b > 0:
|   |   step := step + 1
|   |   if a >= b: a := a mod b
|   |   else:      b := b mod a
|   result := a + b
```

Unfortunately, the log was unfinished. Now Vasya needs a program to calculate the maximal possible length of the log. Obviously, that length is equal to the total number of euclid-maximal pairs with respect to given X.

Input

The input consists of one or more test cases. Each test case consists of a single line with a single integer X — the limit for numbers a and b ($1 \leqslant X \leqslant 10^9$).

The input will be terminated with a test case with $X = 0$, which should not be processed.

The total number of test cases will not exceed 10 000.

Output

For each test case write a single line with a single integer — the number of different pairs.

Example

euclid.in	euclid.out
1 2 3 4 5 10 0	1 3 1 2 1 1

Problem D. Game (Division 1 Only!)

Input file:	game.in
Output file:	game.out
Time limit:	2 seconds (3 for Java)
Memory limit:	256 Mebibytes

Vasya works for RIPSaW (Research Institute of Piles of Stones and Winners). He is studying games like Nim.

Consider the following game for two players. There are n piles of stones. The number of stones in i-th ($1 \leqslant i \leqslant n$) pile equals a_i. On each move, current player can choose pile i and take $1 \leqslant k \leqslant \frac{n}{2}$ stones from it. The player to take the last stone loses.

For simplicity, Vasya decided to change the rules a bit. Now there's one more pile with a_0 stones in it. And the number of stones k that can be taken from it is limited only by its size: $1 \leqslant k \leqslant a_0$.

Given n and $a_1, a_2, \ldots, a_n$, find a_0 to force the first player to lose, assuming both players play optimally.

Input

The input consists of one or more test cases. Each test case consists of a single line containing three integers n, a_1 and d — the number of piles, size of the first pile and difference in sizes of piles i and $(i-1)$ for $1 < i \leqslant n$, respectively. So, pile i will contain $a_1 + d \cdot (i-1)$ stones. It is guaranteed that $n \geqslant 2$ and $1 \leqslant a_1 \leqslant a_n \leqslant 10^{18}$.

The sum of n among all the test cases will never exceed 1 000 000.

Output

For each test case write one line containing the smallest non-negative size of pile 0 such that the game will force the first player to lose. If there is no such size, write -1 instead.

Example

game.in	game.out
2 3 1 3 3 1 4 1 2	0 1 2

Problem E. Logins (Division 1 Only!)

Input file:	`logins.in`
Output file:	`logins.out`
Time limit:	5 seconds (6 for Java)
Memory limit:	256 Mebibytes

... where α is n-th letter of Latin alphabet

from a problem statement

Vasya works for RIPLY (Research Institute of Passwords and Logins Yield). He has written a number of programs that generate logins and passwords based on users' personal data. However, these logins are not always unique. To avoid this problem, Vasya decided to append some strings to each of the logins, making all of them different.

To make the logins closer to initial ones, he wants to keep the maximal length of an appended string as small as possible.

All logins, both initial and resulting, must consist of letters '`a`' and '`b`'.

Input

The input consists of one or more test cases. Each test case starts with a line containing a single integer n — the number of logins. The following n lines contain non-empty strings consisting of letters '`a`' and '`b`' — the initial logins.

The input will be terminated with a test case with $n = 0$, which should not be processed.

The sum of lengths of all logins in the whole input doesn't exceed $5 \cdot 10^5$.

Output

For each test case, write a single line containing the smallest possible maximal length of an appended string.

Example

logins.in	logins.out
2 a a 2 a b 5 a a a a ab 0	1 0 2

Problem F. Orders (Division 1 Only!)

Input file:	`orders.in`
Output file:	`orders.out`
Time limit:	5 seconds (6 for Java)
Memory limit:	256 Mebibytes

Vasya works for RIGHT (Research Institute of Government using Hash Tables). He is studying orders of some government of a country far far away.

In that country all towns are placed along some road. They are also numbered in the order of traversal. Initially, the quality level of life (QLoL) in every town equals zero.

After it, several orders have been issued. There is the only kind of orders: "the QLoL of all the towns i through j must become at least x".

There are also some official statements. They come in the following form: "average QLoL of towns i through j equals x". Vasya wants you to help him check each of these statements in the following way: for each of them, you will be given only the pair (i, j), and your answer must contain the correct value of x.

You may assume that each order was executed, and at each moment of time, each town had the least possible QLoL satisfying all orders.

Input

The input consists of one or more test cases. Each test case starts with a line containing two integers n and k — the number of towns in the country and the number of orders. The following k lines will contain one event each. The event can be one of the following:

1. `^` i j x is an order: after it, all towns numbered i through j must have QLoL at least x ($1 \leqslant x \leqslant 10^9$, $1 \leqslant i \leqslant j \leqslant n$).
2. `?` i j is an official statement; you should calculate the average QLoL for all towns numbered i through j ($1 \leqslant i \leqslant j \leqslant n$).

The input will be terminated with a test case with $n = k = 0$ which should not be processed.

The sum of n's in the whole input will not exceed 100 000. The sum of k's in the whole input will not exceed 100 000.

Output

For each official statement, write a single line with average QLoL formatted as an irreducible fraction with smallest possible natural denominator. If the denominator of some value is 1, write it as an integer instead. See sample output for details.

Example

orders.in	orders.out
10 10 ? 1 10 ^ 1 10 1 ? 1 10 ^ 2 3 10 ^ 3 4 5 ? 2 2 ? 3 3 ? 4 4 ? 1 5 ? 1 10 0 0	0 1 10 10 5 27/5 16/5

Problem G. Data Packets (Division 1 Only!)

Input file:	`packets.in`
Output file:	`packets.out`
Time limit:	2 seconds (3 for Java)
Memory limit:	256 Mebibytes

Vasya works for RIDDLE (Research Institute of Data & Datagram probLEms). He is studying a new protocol of data exchange — an extension for RFC 1149 (IPoAC) and RFC 2549 (IPoAC with QoS). This extension has greater resistance for packet loss, however, another problem arises: two packets in a sequence might get mixed.

To simplify things, consider a pair of packets each containing a single integer — the number of the packet in the packet sequence. These packets were mixed during the transmission, and now instead of two numbers, there is a single sequence of digits. Vasya is sure that the difference between the numbers of packets is small, so he needs a program that can split the string into two numbers with the smallest possible difference.

Input

The input consists of one or more test cases. Each test case consists of a single line of $2 \leqslant n \leqslant 50$ digits 1 through 9.

The sum of lengths of all strings in the whole input doesn't exceed 100.

Output

For each test case write the only line containing two integers with the smallest possible difference which can be mixed, keeping internal order of digits, to obtain the input string. If there are multiple answers, output any of them.

Example

packets.in	packets.out
11 129 23917	1 1 12 9 231 97

Problem H. Regular Convex Polyhedra for Dummies

Input file:	`regular.in`
Output file:	`regular.out`
Time limit:	2 seconds (3 for Java)
Memory limit:	256 Mebibytes

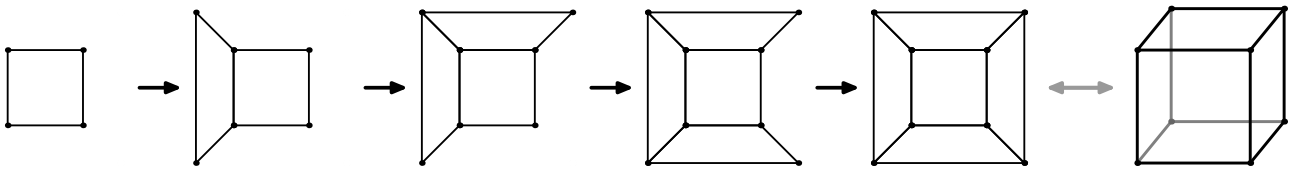
Vasya works for RID (Research Institute for Dummies). He develops new advanced techniques of teaching dummies. His current field of study is geometry for dummies, and the aim is to make complex geometric concepts as simple as possible. Vasya recently finished his work titled "Polygons for Dummies", and is now looking forward to more advanced topics. Unfortunately, his research shows that most dummies have poorly developed three-dimensional imagination, so teaching them 3-D geometry will likely turn into a challenge.

Vasya decided to start the course by introducing the graphically appealing concept of regular convex polyhedra. Recall that a *regular convex polyhedron* is a convex polyhedron with all faces being equal regular polygons. In order to make things simple, Vasya makes a few preliminary observations before actually showing all regular convex polyhedra.

First observation is that each regular convex polyhedron can be characterized by two numbers: p — the number of vertices on each face and q — the number of faces coinciding in each vertex (this number is the same for every vertex of a regular convex polyhedron). For example, a cube has square faces, so $p = 4$, and each vertex of a cube has exactly three faces coinciding in it, so $q = 3$.

Another observation Vasya gives is the fact that the sum of polygon angles at each vertex of a polyhedron must be strictly less than 360^o. For example, there is no regular polyhedron with $p = 6$ and $q = 4$, because a regular hexagon has an angle of 120^o, so the sum of polygon angles in a vertex would be $120 \cdot 4 = 480^o$.

The last observation is that one can easily determine the number of vertices, edges and faces of a regular convex polyhedron without actually working in three dimensions. Suppose we know p and q; draw a graph on the plane starting with a single p-gon. Try adding other p-gons in such a way that all vertices have degree not more than q, and as many vertices as possible have degree of exactly q. This process is illustrated below for $p = 4$ and $q = 3$.



If a polyhedron with the given p and q exists, eventually you will get a graph with all vertices having degree q; all polyhedron's faces were added as p-gons except the last one which is the outer area of the plane — it turns out to have exactly p vertices as well. Note that since we draw the graph on a plane, not in three dimensions, the faces need not be regular p-gons (they even need not be convex). Still, the number of vertices, edges and faces in this planar graph is the same as in the polyhedron.

Vasya claims that these observations are enough to determine for each pair p, q whether a regular convex polyhedron with such characteristics exists, and if it does, how many vertices, edges and faces it has. Write a program which would do that for him.

Input

The input consists of one or more test cases. Each test case consists of a single line with two integers p and q on it — the number of vertices of a regular polygon and the number of polygon intersecting at each vertex, respectively ($3 \leqslant p, q \leqslant 100$).

The input will be terminated with a test case with $p = q = 0$, which should not be processed.

The total number of test cases will not exceed 1000.

Output

For each test case, write a single line with three integers on it — the number of vertices, edges and faces of the respective regular convex polyhedron. If there is no such polyhedron, write three −1s instead.

Example

regular.in	regular.out
4 3 6 4 0 0	8 12 6 -1 -1 -1

Problem I. Router (Division 1 Only!)

Input file:	`router.in`
Output file:	`router.out`
Time limit:	5 seconds (6 for Java)
Memory limit:	256 Mebibytes

Vasya works for RIXXX (Research Institute of Top Secrets). However, that does not matter. He is a system engineer.

He is dealing with a new router and a number of cables he can connect to it. There are n data cables from other devices and n sockets on the router; cable i could be plugged only into socket i of the router. Additionally, each cable has two different plugs: a type A plug and a type B plug; a cable can be either connected to the router by type A plug, connected by type B plug, or not connected at all. If cable i is connected to the router by type A plug, it's network capacity is a_i; if it is connected by type B plug, network capacity is b_i. If a cable is not connected, it's network capacity is zero.

However, there is one problem: type A and type B plugs have different shapes, and the router sockets are tightly packed. As a result, not all plugs could be neighbors: physical limitations don't allow to plug cable i by type A plug and any of the cables $(i+1)$ and $(i+2)$ by type B plug simultaneously. Sockets are arranged in a circle, so you may assume that numbers are cyclic, i. e. cable n is followed by cables 1, 2 and so on.

Vasya's task is to plug in some cables to maximize the sum of network capacities. Help him find an optimal solution.

Input

The input consists of one or more test cases. Each test case starts with a single line containing a single integer n — the number of cables ($4 \leqslant n \leqslant 10^5$). The second line of a test case contains n numbers $a_1, a_2, \ldots, a_n$ — network capacities for type A plugs. The third line of a test case contains n numbers $b_1, b_2, \ldots, b_n$ — network capacities for type B plugs. It is guaranteed that $0 \leqslant a_i, b_i \leqslant 10^9$.

The input will be terminated with a test case with $n = 0$, which should not be processed.

The total sum of n's in all test cases will not exceed 10^5.

Output

For each test case, write a single line with a single integer — the maximal possible sum of network capacities.

Example

router.in	router.out
4 1 2 3 4 4 3 2 1 4 1 0 1 0 0 1 0 1 4 1 6 0 9 1 2 9 5 0	10 2 18

Problem J. Segments

Input file:	segments.in
Output file:	segments.out
Time limit:	2 seconds (3 for Java)
Memory limit:	256 Mebibytes

Vasya works for RISING (Research Institute of Segments IntersectiNG each other). For years he is studying the following problem: given two segments, find their intersection.

As you do not have years to spend during this contest, your task will be a bit easier. Given two segments, each either vertical or horizontal, find their intersection.

Input

The input consists of one or more test cases. Each test case consists of a single line containing eight integers $x_{1,1}, y_{1,1}, x_{1,2}, y_{1,2}, x_{2,1}, y_{2,1}, x_{2,2}, y_{2,2}$. The first four numbers describe endpoints of the first segment, the last four — those of the second one. None of the segments will be degenerate. Each segment is guaranteed to be either horizontal or vertical. Coordinates of points will not exceed 10^9 by absolute value.

The input will be terminated with a test case with $x_{1,1} = y_{1,1} = x_{1,2} = y_{1,2} = x_{2,1} = y_{2,1} = x_{2,2} = y_{2,2} = 0$, which should not be processed.

The number of test cases will not exceed 10 000.

Output

For each test case, write a single line containing one of the following:

- integer 0, if the intersection is empty,
- integer 1, followed by a pair of integers describing a point of intersection, if the intersection consists of a single point,
- integer 2, followed by two pairs of integers describing endpoints of a segment of intersection, if the intersection is a segment. First pair of integers should have lesser sum than second pair.

Example

segments.in	segments.out
0 0 2 0 3 3 0 3 -1 0 1 0 0 -1 0 1 -1 0 1 0 1 0 3 0 -1 0 1 0 0 0 3 0 0 0 0 0 0 0 0 0	0 1 0 0 1 1 0 2 0 0 1 0

Problem K. Team

Input file:	team.in
Output file:	team.out
Time limit:	5 seconds (6 for Java)
Memory limit:	256 Mebibytes

Vasya works for RIP (Research Institute of Psychology, not to be confused with R.I.P. for Research Institute of Primes).

Vasya is carrying out the following experiment. People sit on chairs forming a circle. An integer value is assigned to each of them. After that, they should select a non-empty team of no more than k people sitting in a line with some sum of values. The team members will get some bonus only if their total sum of values will be close enough to the maximal possible sum.

To help Vasya check the results of the experiment, write a program that will calculate the maximal possible sum.

Input

The input consists of one or more test cases. Each test case starts with a single line containing two integers n and k — the number of people and the maximal size of the team ($1 \leqslant k \leqslant n \leqslant 500\,000$). The second line of a test case contains n integer numbers a_i — the values assigned ($|a_i| \leqslant 10^6$).

The input will be terminated with a test case with $n = k = 0$, which should not be processed.

The total sum of n's in all test cases will not exceed 500 000.

Output

For each test case, write a single line with a single integer — the maximal possible sum of values in a team.

Example

team.in	team.out
5 3 9 -9 3 2 0 9 4 3 4 -8 2 -3 7 5 -6 1 0 0	11 12

Problem L. Union

Input file:	`union.in`
Output file:	`union.out`
Time limit:	2 seconds (3 for Java)
Memory limit:	256 Mebibytes

Vasya works for RIPPLE (Research Institute of Politics and PeopLEs). Vasya studies the way country unions are created.

Now he is checking the following model. Consider a set X of countries. Some of them are interested in other ones. That might include trade, border, military or any other interest. The relation of interest is not required to be symmetric.

Consider some particular country a. Vasya calls set $U \subset X$ a *stable union* for $a \notin U$ iff:

- for each country $b \in U$ there is such a sequence of countries $a = t_1, t_2, \ldots, t_k = b$ that for each $1 < i \leqslant k$:
  - country $t_i \in U$ and
  - country t_{i-1} is interested in country t_i

  and
- there is no country $z \notin U \cup \{a\}$ that is interested in some country $q \in U$.

Help Vasya to find the maximal possible stable union for country with number 1.

Input

The input consists of one or more test cases. Each test case consists of a single line containing two integers n and k — the number of countries and the number of "interest pairs" ($1 \leqslant n \leqslant 10^5$, $0 \leqslant k \leqslant 10^5$). The following k lines contain "interest pairs" (a_i, b_i) which mean "country a_i is interested in country b_i" ($1 \leqslant a_i, b_i \leqslant n$, $a_i \neq b_i$).

The input will be terminated with a test case with $n = k = 0$, which should not be processed.

The total sum of n's in all test cases will not exceed 10^5. The total sum of k's in all test cases will not exceed 10^5.

Output

For each test case write a single line with a single integer — the maximal possible size of a stable union for country with number 1.

Example

union.in	union.out
4 3 1 2 2 3 4 3 0 0	1

Problem M. Postman Joe (Division 2 Only!)

Input file:	`postman.in`
Output file:	`postman.out`
Time limit:	2 seconds
Memory limit:	64 Mebibytes

Postman Joe is a fitness fanatic. He has a single row of 20 houses on his delivery route, and sets himself a task every day that will require him to walk up and down the row several times. A typical task will look something like this:

```
3 U3 D2 U1 U6 D4 D5 U7 U9 D5 D3 U5 U4 D2 D5 U8
```

This means that Joe delivers firstly to house 3, moves 3 houses up the street and delivers to house 6, then moves 2 houses down the street and delivers to house 4, and so on. The houses are numbered sequentially from 1 to 20, house 1 being at the bottom of the street. Not every house will necessarily receive a delivery each day, but Joe's instructions must never take him to the same house twice, nor take him beyond either end of the row of houses. To solve this problem, you must write a program to help Joe. It must check that Joe has set himself a legal task and, if so, must report those houses which do not receive a delivery.

Input

Each test represents the deliveries for a single day. Test starts with a single integer S ($1 \leqslant S \leqslant 20$) being the first house to which Joe makes a delivery. This is followed by a sequence of not more than 21 letter-number pairs, each separated by a single space. The letter will be `'U'` or `'D'`, `'U'` meaning go up the street (increasing house number), `'D'` go down the street. The number will be a single digit integer, stating the number of houses to move.

Output

If the instructions for the postman are legal (i.e. no house is visited twice, and Joe is not taken beyond either end of the street), output will be a list of houses that do not receive a delivery that day. The list will be in numerical order with a space between each house number.

If all houses receive a delivery, output should be the word `'none'`. If the instructions for a task are not legal, output should be the word `'illegal'`.

Examples

postman.in	postman.out
3 U3 D2 U1 U6 D4 D5 U7 U9 D5 D3 U5 U4 D2 D5 U8	1 8 14 16
8 D7 U5 U6 D9 U2 D4 U9 U3 D2 U7 U2	illegal

Problem N. DVD (Division 2 Only!)

Input file:	dvd.in
Output file:	dvd.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

A local company, «DVDs R Us», need your help with a stock management system. They sell DVDs on-line from a local warehouse and need to know at any moment how many DVDs of each title they have in stock.

DVDs of a particular title have a stock code and are allocated a certain amount of storeage space in the warehouse. The more popular a DVD title is, the more space is allocated. DVDs are continually being sold and replaced, hence the need for a system to keep track of how many are in stock.

Input

Input consists of data for a DVD title and begins with a stock code, a 7 character code consisting of upper case letters and digits only. The next line for each title consists of two integers, M and S, separated by a space. M is the maximum number of DVDs of that title that can be held in stock at any one time ($20 \leqslant M \leqslant 500$). S is the number of that title currently in stock ($0 \leqslant S \leqslant M$).

The third line is a single integer, T, the number of transactions to follow ($0 \leqslant T \leqslant 100$). There then follow T lines, each containing details of 1 transaction. A transaction line consists of a single upper case letter (`'S'` or `'R'`), followed by a space, followed by a positive integer less than 1000.

- If the letter is `'S'` it represents a sale, in which case the number shows how many of the DVD have been sold. If the sale is for more than the current stock, only those DVDs currently in stock may be sold.
- If the letter is `'R'` it is a restock, so the number shows how many DVDs are added to the current stock. If this would take the number of DVDs in stock to more than the maximum, then the extra items have to be sent back.

Output

Output consists of the DVD's stock code, followed by a space, followed by the number in stock at the end of all the transactions.

Example

dvd.in	dvd.out
HG67966 100 64 3 S 10 S 25 R 30	HG67966 59

Problem O. Arithmetic Sequence (Division 2 Only)

Input file:	`seq.in`
Output file:	`seq.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

An arithmetic sequence is one in which there is some first number, and then a series of numbers which are all a fixed number different. For example 3, 5, 7, 9 is an arithmetic sequence that has a first number 3. Then each term after that in the sequence is formed by adding 2 to the previous term. (The terms are different by 2). The 3 is also called the first term (term 1) and 9 is the 4th term. Given a starting number, a difference and a value, your program is to work out if the number could be part of the sequence. If so, output which term that number would be, and if not, output a letter `'X'`.

Input

Input file has one line with 3 numbers separated by spaces. The first number a_1 is an integer that is the first term in the sequence. The second is the difference — this will be a non-zero integer k. The third is the value x that you will need to test to determine whether it can be part of the sequence or not ($-6 \cdot 10^6 \leqslant a_1, k, x \leqslant 6 \cdot 10^6$).

Output

Output will consist of either a number indicating which term it is, or `'X'` if the number isn't part of the sequence.

Example

seq.in	seq.out
3 2 11	5
-1 -3 -8	X

Problem P. Extreme Tic Tac Toe (Division 2 Only)

Input file:	ettt.in
Output file:	ettt.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Tic Tac Toe is a simple pencil and paper game played by children. The idea is that players take turns at drawing their symbol (‘O’ or ‘X’) in squarex on a 3 by 3 grid. The first to get three of their symbols in a row (horisontal, vertical or diagonal) wins the game. If the grid is filled withoit either player making a line, the game is a draw.

In game of Extreme Tic Tac Toe (ETTT) used N-dimension board (built as 3 layers of a $N-1$-dimension board; 2-dimension board is board for plain Tic Tac Toe). ETTT scoring differs from normal Tic Tac Toe. Rather than stop as soon as one player achieves a line of their symbols, ETTT players play until the grid is full, or until they agree to stop. The winner is the one with the greatest number of lines of symbols.

Your task is to write a program to read N-dimensional ETTT game grids and work out the scores.

Input

The input starts with a line holding N ($2 \leqslant N \leqslant 10$) — the dimension of game. That is followed by lines of ‘X’, ‘O’, and ‘.’ for the cells of game. Each line holds at least one, and no more than 40 symbols. To understand the order in which data is input, imagine the board being held in an N-dimensional array. With $N = 5$, for example, the board could accessed as cell[a,b,c,d,e] (or cell[a][b][c][d][e]). The following pseudo-code would read data in the correct order (ignoring line breaks)

```
for a = 1 to 3
   for b = 1 to 3
         for c = 1 to 3
                  for d = 1 to 3
                            for e = 1 to 3
                                   read cell[a,b,c,d,e]
```

Output

Output the scores for ‘X’ and ‘O’, as shown in the sample data.

Example

ettt.in	ettt.out
2 XXX OO. X.O	X scores 1 and O scores 0
3 XXX ..O .OO X...X...X X.O .OO O.X	X scores 4 and O scores 1