Problem A. Average

Input file:	`average.in`
Output file:	`average.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

You have acquired a list of the math and verbal test scores from all the children in the county. Write a program that takes the math and verbal scores of all of the children in the county, and finds the number of children who have a composite score which is below average in the county.

The composite score is defined to be the sum of a child's math and verbal scores.

Input

The first line contains the number of children N which is between 1 and 50 elements, inclusive. Two lines follow, each containing N numbers — the math and verbal scores, respectivile. The scores are between 200 and 800, inclusive.

Output

Output the number of children who have a composite score which is below average in the county.

Examples

average.in	average.out
4 200 250 700 700 400 400 400 400	2
2 500 400 300 400	0
1 293 799	0
7 400 400 400 400 400 400 401 400 400 400 400 400 400 400	6

Problem B. Bits

Input file:	`bits.in`
Output file:	`bits.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Computers operate on binary numbers. Almost all computation is done by manipulating 0's and 1's. Thus, in order for computers to use the numbers we give them, they must convert them from base 10 (what we use normally) into binary (base 2). It is sometimes useful to determine how many bits a number will take to represent, so that we can save memory. For example, if a number is smaller than 256, we can represent it with 8 bits.

A binary number's value is determined as follows: For each '1' in the binary number add 2^i (2 to the power of i), where i is the number of digits to the right of the '1'. For example, 10100 binary, is equivalent to 20 in decimal. The first 1 has 4 digits to the right, so it is equivalent to $2^4 = 16$. The other 1 has two digits to the right of it, so it is $2^2 = 4$. $16 + 4 = 20$. Another example is 1111, whose base 10 equivalent is $2^3 + 2^2 + 2^1 + 2^0 = 8 + 4 + 2 + 1 = 15$.

Your task is to write a program that will determine the minimum number of bits that must be used to represent the given number in binary.

Input

Just one integer n which is between 1 and 1 000 000, inclusive.

Output

Output the minimum number of bits that must be used to represent the given number in binary.

Examples

bits.in	bits.out
32	6
12	4
1	1
1500	11

Problem C. Chivalry

Input file:	`chivalry.in`
Output file:	`chivalry.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Two lines (or queues) of people often have to merge into a single-file line. But, chivalry is not dead! When a man and a woman are about to merge, the man will always let the woman go first.

Given two lines of both men and women, write a method getOrder which will determine the genders of the people in the final line. If two women are at the front of both lines, the woman from the first line goes first. Likewise, if two men are at the front of both lines, the man from the first line goes first. Then, the people at the front of both lines are compared again.

Input

The input contains two lines. Each input line will be a string of letters, each one representing either a man or a woman. Each man will be represented by an '`M`' and each woman by a '`W`'. Both lines will be between 1 and 50 characters long, inclusive.

Output

The output should be of the same form. The front of each line is the left-most character of the string.

Examples

chivalry.in	chivalry.out
M W	WM
MM MW	MMMW
MMMM W	WMMMM
M WWW	WWWM

Problem D. Dither Counter

Input file:	`dither.in`
Output file:	`dither.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Sometimes when computer programs have a limited number of colors to use, they use a technique called dithering. Dithering is when you use a pattern made up of different colors such that when the colors are viewed together, they appear like another color. For example,

you can use a checkerboard pattern of black and white pixels to achieve the illusion of gray.

You are writing a program to determine how much of the screen is covered by a certain dithered color. Given a computer screen with each pixel being a certain color, and a list of all the solid colors that make up the dithered color, find the number of pixels on the screen that are used to make up the dithered color. Each pixel will be represented by a character in screen. Each character in screen and in dithered will be an uppercase letter ('A'–'Z') representing a color.

Assume that any pixel which is a color contained in dithered is part of the dithered color.

Input

The first line contains two integers N and M representing the dimensions of the screen, followed by a description of dithered color which will contain between 2 and 26 upper case letters ('A'–'Z'), inclusive. There will be no repeated characters in the description. N and M will be between 1 and 50, inclusive. N lines follow, each containing M upper case letters ('A'–'Z').

Output

Output the number of pixels on the screen that are used to make up the dithered color.

Examples

dither.in	dither.out
6 8 BW AAAAAAAA ABWBWBWA AWBWBWBA ABWBWBWA AWBWBWBA AAAAAAAA	24
6 8 BW BBBBBBBB BBWBWBWB BWBWBWBB BBWBWBWB BWBWBWBB BBBBBBBB	48
3 7 CA BBBBBBB BBBBBBB BBBBBBB	0
1 4 DCBA ACBD	4

Problem E. Measures

Input file:	measures.in
Output file:	measures.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

We are producing music analysis software. We have already broken the sensor data into a sequence of beats, with the loudness of each beat known.Your project is to divide the beats into measures. Each measure consists of adjacent beats, and each beat must be in exactly one measure. The number of beats per measure must be between 2 and 10, inclusive, and be the same for all measures.

The heuristic that has been chosen is to divide the beats into measures based upon the idea that the first beat in each measure tends to be stressed. The data does not necessarily begin or end with a full measure, so if you decide on k beats per measure then the first full measure may begin at any of the first k beats in the loudness data. The primary requirement is that at least 80% of the full measures must have the loudness of their first beat at least as big as the loudness of each of the other beats in that measure. It is also required that there must be at least one full measure.

Given the loudness of each beat, write a program which finds the smallest acceptable number of beats per measure.

Input

The input contains of 2 to 50 numbers — the loudness of each beat. The numbers will be between 0 and 100, inclusive.

Output

Output the smallest acceptable number of beats per measure. If there is no value between 2 and 10 inclusive satisfies the requirements, output −1.

Examples

measures.in
3 9 3 0 9 3 1 9 11
measures.out
3

measures.in
3 9 3 0 9 3 1 9 11 1
measures.out
5

measures.in
5 2 5 2 8 6 3 5 5 1 9 0 2
measures.out
2

measures.in
1 6 9
measures.out
-1

Problem F. Medici

Input file:	medici.in
Output file:	medici.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

In the new boardgame Medici, you play an aristocrat in 15th-century Florence. On each turn, you gather points in three categories: fame, fortune, and dirty secrets with which you can blackmail other players. The winner is the player with the most points in his weakest category at the end of the game. For example, suppose there are three players with the following points:

NAME	FAME	FORTUNE	SECRETS
Salvestro	20	60	40
Giovanni	30	80	30
Cosimo	50	40	50

Salvestro's weakest category is Fame, in which he has 20 points. Giovanni's weakest categories are Fame and Secrets, with 30 points in both. Cosimo's weakest category is Fortune, with 40 points. Cosimo is the winner.

You will be given the values for fame, fortune, and secrets. You are to find the (zero-based) index of the winning player. If there is a tie for the winning score, the game is declared a draw and replayed, in which case you should output −1.

Input

The first line contains the number of players N, which is between 2 and 20 elements, inclusive. Three lines of N numbers follow, each containing N values of fame, fortune and secrets for each player. All values of fame, fortune, and secrets are between 0 and 100, inclusive.

Output

Output the index of the winning player or -1.

Examples

medici.in	medici.out
3 20 30 50 60 80 40 40 30 50	2
4 0 100 100 100 100 0 100 50 50 50 50 100	-1
2 1 0 1 99 1 99	0
3 39 42 57 42 39 57 57 39 42	2

Problem G. Moving Averages

Input file:	moving.in
Output file:	moving.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

Moving averages are well known in stock charts analysis. They are used to emphasize the direction of a trend and to smooth out fluctuations. Athletes may use moving averages to analyze their training results.

Given the times from successive training sessions (e.g. the time to cycle a certain leg) and the integer value of n, find the n-moving averages in seconds for these times, with each average rounded down.

A n-moving average is the average (i.e. the arithmetic mean) of n consecutive times. So for t times given, $t-n+1$ n-moving averages are to be calculated. The first average is composed from the times 1 to n, the second average from the times 2 to $n+1$ and so on, the last average is composed from the times $t-n+1$ to t.

Input

The first line contains the number t which is between 1 and 50 elements, inclusive. t lines follow, each in the format "hh:mm:ss" (quotes for clarity), where hh, mm and ss are two digit numbers (with a leading zero if necessary) indicating the number of hours (between 0 and 23, inclusive), minutes and seconds (between 0 and 59, inclusive), respectively. The last line contains the number n which is between 1 and t, inclusive.

Output

Output the n-moving averages in seconds for these times, with each average rounded down.

Examples

moving.in	moving.out
5 01:19:10 01:17:40 01:19:44 01:17:23 01:17:07 3	4731 4695 4684
5 01:19:10 01:17:40 01:19:44 01:17:23 01:17:07 1	4750 4660 4784 4643 4627
5 01:19:10 01:17:40 01:19:44 01:17:23 01:17:07 5	4692

Problem H. Multiples

Input file:	multiples.in
Output file:	multiples.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

Given a range of integers from A to B (inclusive), determine how many numbers within that range are evenly divisible by C.

Note

If x is evenly divisble by y, there exists some integer k such that $ky = x$.

Input

Three integers: A, B and C. A and B do not exceed 10^6 by an absolute value. C is between 1 and 1000, inclusive.

Output

The answer to the task.

Examples

multiples.in	multiples.out
0 14 5	3
7 24 3	6
-123456 654321 997	780
-75312 407891 14	34515

Problem I. No Order Of Operations

Input file:	noorder.in
Output file:	noorder.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

When evaluating a mathematical expression, there is the possibility of ambiguity. If you wanted to know the result of "`3 + 5 * 7`", you might first evaluate the `(3+5)` and get 56, or first evaluate the `(5*7)` and get 38. This ambiguity can be resolved by using the order of operations: first do multiplication and division (from left to right),

and then after all those are done, do addition and subtraction (again from left to right). Here, the correct result would be the 38.

While this is unambiguous, it certainly is somewhat annoying. You think it would be easier if people did all math from left to right, all the time, and want to make a simple expression evaluator to do so.

Input

There is only one line in the input file containing the expression given to you. It will consist of one digit numbers (`0` through `9`) alternating with operators (`+`, `-`, or `*`), with no spaces between them. Thus, expression would follow the format *Digit Operator Digit Operator ... Digit*. For example, the expression given above would be given as "`3+5*7`". The length of expression does not exceed 17 characters. The expression can't be empty.

Output

Your should output the value of the expression when evaluated from left to right.

Examples

noorder.in	noorder.out
3+5*7	56
4-8*9*1	-36
0	0
1*2*3*4*5*6*7*8*9	362880

Problem J. Letter Strings

Input file:	strings.in
Output file:	strings.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

A letter-string is composed of letters ('`A`'–'`Z`','`a`'–'`z`') and dashes ('`-`'). The length of a letter-string is the number of characters in it not including dashes (in other words, the number of letters in the string). Given a list of letter-strings you will return the sum of their lengths.

Input

The input will consist of 1 to 50 strings, each of them will have length between 1 and 50, inclusive. Each string will contain only letters ('`A`'–'`Z`','`a`'–'`z`') and dashes ('`-`').

Output

The answer to the task.

Example

strings.in	strings.out
-	0
A	1
-----Abc	3
-A-B-C-D --------EFGHI JKLMNOPQR ---STU-VW-XYZ	26

Problem K. Transport Counting

Input file:	transport.in
Output file:	transport.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

You are studying public transportation, and you want to know how many buses are going down a particular one-way street every minute. You are driving along the street by car, and counting the buses you meet or overtake. After some time, you stop counting and report the result. In this problem, you may assume that the street is a straight line, and that your car and all of the buses can only go along this line in the same direction.

You will be given your speed in meters per minute. You will also be given positions, specifying how far ahead of you each of the buses is in meters at time 0, and velocities, specifying the velocities of the buses in meters per minute. Also you will be given K, the number of minutes you should count the buses you pass for.

You should find the number of buses you will overtake or meet during K minutes. If you meet one or several buses at the first or at the final moment, count them also.

Input

The first line contains three integers: S, K and N, where S is your speed and N is the number of buses. S and K will both be between 0 and 1000 inclusive, and N is between 0 and 50 elements, inclusive.

Two lines follow, containing N numbers each — positions and velocities of buses, respectively. Positions and velocities will be between 0 and 1000, inclusive.

Output

Output the number of buses you will overtake or meet.

Examples

transport.in	transport.out
100 0 1 0 0	1
5 2 2 10 10 0 1	1
5 3 2 10 10 0 1	2
5 10 3 0 0 0 4 5 6	3

Problem L. Username

Input file:	username.in
Output file:	username.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

You are implementing the member registration system of an online dating site. When a new member signs up, it is possible that she initially chooses the same username as an existing member. The system must then inform the new member of the conflict and suggest a variant of the chosen name with a number attached to the end.

If an existing member is named "`FunkyMonkey`", for example, and a new member wants the same username, the simplest suggestion the system can make is "`FunkyMonkey1`". If there is already a member by that name, the system must suggest "`FunkyMonkey2`", unless that variant is also taken. If all names from "`FunkyMonkey1`" through "`FunkyMonkey9`" are taken as well as the original "`FunkyMonkey`", the system moves on to consider "`FunkyMonkey10`", and so on. The goal is to use the smallest possible number in the variant. Note that each username consists of letters (the characters from '`a`' to '`z`' and from '`A`' to '`Z`') and numerals ('`0`' to '`9`').

In the event of a conflict, this member will accept the suggestion of-

fered by your system in accordance with the principles above. Your task is to find the username finally assigned to the new member.

Input

The first line of the input file contains the number of existing names N, which is between 1 and 50, inclusive. N lines follow, each containing between 1 and 50 characters long, inclusive. The only characters permitted in existing names are '`a`' to '`z`', '`A`' to '`Z`', and '`0`' to '`9`'. There are no existing names which end in a number that has a leading zero. The last line of the input contain the username that a new member wants to use. It consists of 1 to 50 characters (inclusive) '`a`' to '`z`' and '`A`' to '`Z`'.

Output

Output the username finally assigned to the new member.

Note

The constraints rule out names that end in a number with a leading zero, such as "`grokster006`" and "`bart0`".

Examples

username.in	username.out
5 MasterOfDisaster DingBat Orpheus WolfMan MrKnowItAll TygerTyger	TygerTyger
7 MasterOfDisaster TygerTyger1 DingBat Orpheus TygerTyger WolfMan MrKnowItAll TygerTyger	TygerTyger2
7 TygerTyger2000 TygerTyger1 MasterDisaster DingBat Orpheus WolfMan MrKnowItAll TygerTyger	TygerTyger
8 grokster2 BrownEyedBoy Yoop BlueEyedGirl grokster Elemental NightShade Grokster1 grokster	grokster1
16 Bart4 Bart5 Bart6 Bart7 Bart8 Bart9 Bart10 Lisa Marge Homer Bart Bart1 Bart2 Bart3 Bart11 Bart12 Bart	Bart13

Problem M. Yahtzee Score

Input file:	`yahtzee.in`
Output file:	`yahtzee.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

This task is about the scoring in the first phase of the die-game Yahtzee, where five dice are used. The score is determined by the

values on the upward die faces after a roll. The player gets to choose a value, and all dice that show the chosen value are considered active. The score is simply the sum of values on active dice.

Say, for instance, that a player ends up with the die faces showing 2, 2, 3, 5 and 4. Choosing the value two makes the dice showing 2 active and yields a score of $2 + 2 = 4$, while choosing 5 makes the one die showing 5 active, yielding a score of 5.

Your program will take as the information about upward faces of dice, and output the maximum possible score with these values.

Input

Five numbers between 1 and 6, inclusive.

Output

Output the maximum possible score with these values.

Examples

yahtzee.in	yahtzee.out
2 2 3 5 4	5
6 4 1 1 3	6
5 3 5 3 3	10

Problem N. Access Level

Input file:	access.in
Output file:	access.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

In many computer systems and networks, different users are granted different levels of access to different resources. In this case, you are given user rights, indicating the privilege level of each user to use some system resource. You are also given the minimum permission a user must have to use this resource.

You are to find which users can and cannot access this resource.

Input

The input starts from an integer number N followed by a number P. N is the number of users, it is bounded between 0 and 50, inclusive. P is the minimal permission level between 0 and 100, inclusive. N integers follow, indicating user privelege leves. These integers are between 0 and 100, inclusive.

Output

Output a string of N characters. Character '`A`' indicates the user is allowed access, while '`D`' indicates the user is denied access.

Examples

access.in	access.out
6 2 0 1 2 3 4 5	DDAAAA
5 20 5 3 2 10 0	DDDDD
0 20	
6 49 34 78 9 52 11 1	DADADD

Problem O. British Coins

Input file:	british.in
Output file:	british.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

Prior to 1971, Britain used a system of coins that can be traced back to the time of Charlemagne. The three main units of coinage were the penny, the shilling, and the pound. There were 12 pennies in a shilling and 20 shillings in a pound. Given a number of pennies, convert this amount into pounds, shillings, and pennies by first converting as many pennies as possible into pounds, and then converting as many of the remaining pennies as possible into shillings.

Input

The number of pennies which is between 0 and 10000, inclusive.

Output

Output the number of pounds, the number of shillings, and the number of pennies, in that order.

Examples

british.in	british.out
533	2 4 5
0	0 0 0
6	0 0 6
4091	17 0 11
10000	41 13 4

Problem P. Divisor Digits

Input file:	divdigits.in
Output file:	divdigits.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

Write a program which takes an integer number and returns how many digits in number that number itself is divisible by. Count all occurences of such digits in the number, not just the first.

Input

Number will be between 10 000 and 999 999 999, inclusive (between 5 and 9 digits, inclusive).

Examples

divdigits.in	divdigits.out
12345	3
661232	3
52527	0
730000000	0

Problem Q. Equal Substrings

Input file:	equal.in
Output file:	equal.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

You will be given a string consisting of lowercase letters. You will find two strings x and y that must satisfy:

1. The string xy (x with y concatenated on the end) must equal original string.
2. The number of `a`'s in x must equal the number of `b`'s in y.

3. If multiple solutions are possible, use the one that maximizes the length of x.

Input

The only input string will contain between 1 and 50 characters inclusive. Each character in string will be a lowercase letter ('a'–'z').

Output

Output x on the first line, y on the second one.

Examples

equal.in
aaabbb

equal.out
aaa bbb

equal.in
bbbaaa

equal.out
bbb aaa

equal.in
bbbbbb

equal.out
bbbbbb

equal.in
aaaaaa

equal.out
aaaaaa

equal.in
abjlkbjalkbjaljsbljbalb

equal.out
abjlkbjalkbjaljs bljbalb

Problem R. Fuel Consumption

Input file:	fuel.in
Output file:	fuel.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

You are taking your car on a long trip and have only a limited amount of fuel. You know how many liters of fuel your car uses per hour for certain speeds and you'd like to know how far a certain amount of fuel will take you when travelling at the optimal speed.

You will be given velocities and consumptions. Velocities are specified in kilometers per hour. Consumptions are the amounts of fuel (in milliliters) the car will consume in 1 hour, if your speed is equal to the i-th velocity. In addition, you will be given the total amount of fuel in milliliters. Your program should find the maximum distance that the car can travel (in kilometers) with the given amount of fuel, and travelling at a constant velocity equal to one of the elements of velocities.

Input

The first line contains two integers N and F, where N is the number of possible velocities, which will be between 1 and 50 elements, inclusive. F will be between 100 and 50000, inclusive. Two following lines specify velocities and consumptions. Each velocity will be between 5 and 250, inclusive. Each consumption will be between 1000 and 20 000, inclusive. There will be no duplicate velocities.

Output

Output the maximum distance that the car can travel. The maximal absolute error allowed is 10^{-6}.

Examples

fuel.in	fuel.out
1 10000 100 10000	100.0
6 40000 70 80 90 100 60 110 4000 4000 4000 4000 4000 4000	1100.0
6 50000 250 240 230 220 210 211 5000 4500 4000 3500 3000 3000	3516.6666666666665
5 47832 5 10 20 40 80 1000 2500 6250 9000 18000	239.16
6 47832 5 10 20 40 80 160 1000 2500 6250 8000 9500 20000	402.79578947368424

Problem S. Golf Score

Input file:	golf.in
Output file:	golf.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

A full-sized golf course consists of 18 lawns known as holes. The player's objective is to strike a ball with his club in such a way that it travels from a specified point at one end of the lawn to a specified point at the other, and to do so with as few strokes as he can. Associated with each hole is a positive number, the par, which is the number of strokes it is expected to take a competent golfer to complete the hole.

A player's performance on an individual hole is described by a phrase that depends on the number of strokes he took relative to par. To make a "bogey", for example, means that the player has completed a hole in one stroke more than the par value, and a "double bogey" is two strokes over par. Two strokes under par, on the other hand, is an "eagle", while the "albatross", a rare bird indeed, is three strokes under par. The following is a complete dictionary of scoring phrases.

triple bogey	three strokes over par
double bogey	two strokes over par
bogey	one stroke over par
par	exactly par
birdie	one stroke under par
eagle	two strokes under par
albatross	three strokes under par
hole in one	exactly one stroke

The managers of Gravel Mountain Golf Course have contracted you to implement a score-management system that will translate a single player's scores from the above jargon to a numerical total. You are given the par value of each of the course's 18 holes in playing order.

You are also given the player's score on each hole. The reported scores will be valid and complete. Compute the player's total score.

Input

The input file consists of 18 lines. Each line consists of the par value which is between 1 and 5, inclusive, and the score which is one of the eight phrases listed above.

Output

Output the player's total score.

Examples

golf.in	golf.out
1 bogey 1 bogey 1 bogey 1 bogey 1 bogey 1 bogey 1 bogey 1 bogey 1 bogey 5 eagle 5 eagle 5 eagle 5 eagle 5 eagle 5 eagle 5 eagle 5 eagle 5 eagle	45
3 bogey 2 double bogey 4 par 2 double bogey 2 double bogey 1 triple bogey 1 triple bogey 1 triple bogey 3 bogey 2 double bogey 4 par 4 par 4 par 2 double bogey 3 bogey 1 triple bogey 3 bogey 2 double bogey	72

Problem T. Inequality Checker

Input file:	inequality.in
Output file:	inequality.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

Using mathematical induction it is possible to prove the following inequality when $n > 1$:

$$s = 1^3 + 2^3 + ... + (n-1)^3 < n^4/4 < 1^3 + 2^3 + ... + n^3 = S$$

Given n, find $(S + s)/2 - n^4/4$ as a numerator and denominator of fraction when written in least terms (reduced).

Input

The input contains the only integer n which will be between 2 and 100 inclusive.

Output

Output two numbers: the numerator and the denominator.

Examples

inequality.in	inequality.out
2	1 1
3	9 4
100	2500 1

Problem U. Input Box Checker

Input file:	inputbox.in
Output file:	inputbox.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

Imagine a simple input dialog box. The user is supposed to enter a positive integer from a given range into the box.

Recently you realized that sometimes you can tell that the user's input is invalid before he finishes entering the number. For example, if the valid range is 300 to 347, and the user wants to enter the number 372, as soon as he types 37 you can be sure that his input won't be valid.

More precisely, we call a number valid if it is a prefix of some number in the given range, and invalid otherwise.

You are given the range of valid inputs a and b and the input numbers. The range of valid inputs contains all integers between a and b, inclusive.

Write a program that will determine for each number in numbers whether it represents a valid input for the given range.

Input

The first line consists of three integers a, b and N. a and b are between 1 and 2 000 000 000, inclusive. a is less than or equal to b. N is between 1 and 50, inclusive. N input integers follow, each of them is between 1 and 2,000,000,000, inclusive.

Output

For each element of numbers represents a valid input, output `VALID`, otherwise it has to be `INVALID`.

Examples

inputbox.in	inputbox.out
300 347 1 37	INVALID
310 320 6 3 31 317 3174 310 320	VALID VALID VALID INVALID VALID VALID
600 1020 4 7 73 734 7349	VALID VALID VALID INVALID
64 78 9 1 2 3 4 5 6 7 8 9	INVALID INVALID INVALID INVALID INVALID VALID VALID INVALID INVALID

Problem V. Leap Age

Input file:	leap.in
Output file:	leap.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

Your friend was born on "leap day" and has always been troubled by the fact that he just turned 24 this year on his 5th birthday. You would like to write him a nice program for his birthday to help him and other leap-day-born people keep track of how many birthdays they've had so far.

You will be given two years: the current year and the year the person was born. You are to find the number of leap days that have occurred in that time span, not including the first year if it is a leap year, but including the current year if it's a leap year.

A leap year is any year that is a multiple of 4, unless it is divisible by 100 and not 400. For example, 1984 was a leap year, but 1900 wasn't, because it was divisible by 100 and not 400. 2000 was a leap year, because it was divisible by both 100 and 400.

Input

The input contains two integers: the current year and the year the person was born. They will both be between 1582 and 10 000 (1582 is when the Gregorian Calendar started, so there were no leap years before then).

Output

Output the number of leap days that have occurred in that time span.

Examples

leap.in	leap.out
2004 1980	6
10000 1582	2042
2007 1981	6
1981 1980	0
1984 1983	1
9700 5795	947

Problem W. Margin Calculator

Input file:	margin.in
Output file:	margin.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

Margin is defined as the percentage of the selling price of an item or group of items which is profit. For example, if an item costs $80 and is sold for $100, then there is $20 profit, or 20% margin.

You will be given the items sold in a single transaction. Each item is described by two numbers. The first number listed is the price the customer paid for the item. The second number is what the cost to the store of the item was.

You will write a program which will calculate the percentage of margin on the transaction and output it rounded down to the greatest integer less than the actual value.

Note

The cost for some items may be greater than the price they was sold for. However, the sum of the prices of all the items in the transaction will be greater than the sum of the costs of all items.

Input

The input file contains between 1 and 50 strings. Each string will be formatted as follows: "nnn.nn nnn.nn" (quotes for clarity), where each n is a digit between '0' and '9' inclusive. Each string will be exactly 13 characters in length. The total price of all items will be greater than the total cost of all items. The total price of all items will be greater than 0. The percent of margin, prior to rounding, will not be within 10^{-5} of an integer.

Output

Output the percentage of margin on the transaction rounded down to the greatest integer less than the actual value.

Examples

margin.in	margin.out
012.99 008.73 099.99 050.00 123.45 101.07	32
000.00 049.99 999.99 936.22 033.99 025.64 249.99 211.87	4
822.77 704.86 829.42 355.45 887.18 949.38	20

Problem X. Noisy Sensor

Input file:	noisy.in
Output file:	noisy.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

You are processing data obtained from a temperature sensor in a physics experiment. The data consists of an array of integers that represents the measured temperature as a function of time. Unfortunately, the sensor is of poor quality and has introduced some random noise into the data. You have decided to apply a median filter to the temperature data in order to reduce the effect of this noise.

In the context of this problem, median filtering is an operation on an array that replaces each element i except the first and last with the

median value of itself and its two immediate neighbors (the three elements i, $i-1$, and $i+1$). The first and last elements are missing a neighbor, so median filtering does not affect them.

Input

The input consists of 1 to 50 elements, inclusive. Each element is between $-2\,147\,483\,648$ and $2\,147\,483\,647$, inclusive.

Output

Output the median-filtered input.

Examples

noisy.in	noisy.out
1 2 3 4 5 6 7 8 9 10	1 2 3 4 5 6 7 8 9 10
10 1 9 2 8	10 9 2 8 8
500 500 500 500 500	500 500 500 500 500
-2147483648 2147483647	-2147483648 2147483647

Problem Y. String Compare

Input file:	string.in
Output file:	string.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

Your company is writing a spell-checker system, and you have been tasked with writing a function to determine how closely two words resemble each other. The algorithm you are to use, albeit not a very good one, is to compare the two words character by character, and count how many times the characters in a given position are the same. For instance, the words "TICK" and "TOCK" have a score of 3, since three characters (T, C, K) are the same. Similarly, "CAT" and "DOG" score 0, since no letters match.

You are given two strings a and b and are to find the score (as defined above) of how closely the two match.

Input

The input consists of two strings a and b which are between 1 and 50 characters, inclusive. They consists only of letters 'A'–'Z'.

Output

Output the score.

Examples

string.in	string.out
TICK TOCK	3
CAT DOG	0
APPLE APPLES	5
FANTASTIC ANTASTIC	0

Problem Z. Waiter Tipping

Input file:	waiter.in
Output file:	waiter.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

You have just finished eating your Chinese food, and the waiter has brought you the bill. You note the untaxed total on the bill. Additionally, you know the tax rate in your locale. Lastly, you have counted how much money you have.

Since you feel the service was excellent, you want to give as large a tip as you can afford. You are to find the largest integral value of tip such that:

$$total + \text{floor}(total \cdot taxPercent/100) + \text{floor}(total \cdot tip/100) \leqslant money$$

Input

Input consists of three integers: $total$, $taxPercent$ and $money$. $total$ and $money$ will be between between 100 and 100000, inclusive. $taxPercent$ will be between 0 and 100, inclusive.

Output

If there is no non-negative value of tip that satisfies the above inequality, output -1 (you don't have enough money to pay the bill and tax). Otherwise output the largest integral value of tip.

Examples

waiter.in	waiter.out
500 10 600	10
500 10 604	10
850 8 870	-1
226 48 584	111
123 52 696	415
500 10 550	0