Shot Contest 2014

1 de noviembre de 2014

Poli And UN together to drink

Problem Set.

Reglas:

- El first solve acarrea un shot para todos los equipos excepto para el que hizo el first solve.
- Cada tres envíos malos de un equipo en un mismo problema acarrea un shot para el equipo.
- Cada envío aceptado de un problema hace girar la ruleta para sortear shots, si cae el equipo que provocó el giro de la ruleta, se gira nuevamente hasta que caiga otra opción(sólo el primero de cada problema de cada equipo)
- Las reglas pueden cambiar según decida el equipo más importante (el de coaches)

Sitio oficial `http://www.alcoholicos-anonimos.org/`

Síguenos en Twitter `@AAGrupoDar`

A - Sums

Nombre del archivo a enviar: `sums.cpp`, *or* `sums.java`

Given a string of digits, find the minimum number of additions required for the string to equal some target number. Each addition is the equivalent of inserting a plus sign somewhere into the string of digits. After all plus signs are inserted, evaluate the sum as usual. For example, consider the string "12" (quotes for clarity). With zero additions, we can achieve the number 12. If we insert one plus sign into the string, we get "1+2", which evaluates to 3. So, in that case, given "12", a minimum of 1 addition is required to get the number 3. As another example, consider "303" and a target sum of 6. The best strategy is not "3+0+3", but "3+03". You can do this because leading zeros do not change the result.

Write a program that takes the String $numbers$ and an integer sum. The program should calculate and print the minimum number of additions required to create an expression from numbers that evaluates to sum. If this is impossible, print -1.

- $numbers$ will contain between 1 and 10 characters, inclusive.
- Each character in $numbers$ will be a digit.
- sum will be between 0 and 100, inclusive.
- For the first sample test case: In this case, the only way to achieve 45 is to add 9+9+9+9+9. This requires 4 additions.
- For the second test case: Be careful with zeros. 1+1+1+0=3 and requires 3 additions.
- For the fifth test case: There are 3 ways to get 100. They are 38+28+34, 3+8+2+83+4 and 3+82+8+3+4. The minimum required is 2.

Input

The input will contain several test case, each in a line. The String $numbers$ and the integer sum.

Output

Print a line per test case, the solution to the problem.

The output must be written to the standard output.

Input sample	Output sample
99999 45	4
1110 3	3
0123456789 45	8
99999 100	-1
382834 100	2
9230560001 71	4
0000000000 0	0
111 3	2
1111111111 10	9
1212121212 15	9
1213121712 21	9

B - Tower

Nombre del archivo a enviar: `tower.cpp`, *or* `tower.java`

John and Brus are building towers using toy bricks. They have an unlimited supply of bricks of C different colors. Each brick is a 1x1x1 cube. A tower of height X is a 2x2xX rectangular prism, built using 4X bricks.

John and Brus want their towers to look nice. A tower is nice if it has the following two properties:

There are at most K pairs of neighboring bricks with the same color. (Two bricks are neighboring if they share a common side.) The height of the tower is between 1 and H, inclusive. You are given the integers C, K, and H. print the number of nice towers, modulo $1,234,567,891$.

- C will be between 1 and 4, inclusive.
- K will be between 0 and 7, inclusive.
- H will be between 1 and 47, inclusive.
- For the first sample test case: No two neighboring bricks may share the same color. As we only have two colors, the entire tower must be colored like a chessboard. There are two such towers of height 1, and two of height 2.
- For the second test case: Only one tower of height 1 is acceptable here.
- For the third test case: There are 16 possible towers of height 1. If all bricks share the same color, the tower is not nice. There are two such towers. Each of the remaining 14 towers is nice.

Input

The input will contain several test case, each in a line, the integer values C, K and H.

Output

Print a line per test case, the solution to the problem.

The output must be written to the standard output.

Input sample	Output sample
2 0 2	4
1 7 19	3
2 3 1	8
4 7 47	-1
1 3 19	2
3 6 26	4

C - Songs

Nombre del archivo a enviar: `songs.cpp`, *or* `songs.java`

Vocaloids $Gumi$, Ia, and $Mayu$ love singing. They decided to make an album composed of S songs. Each of the S songs must be sung by at least one of the three Vocaloids. It is allowed for some songs to be sung by any two, or even all three Vocaloids at the same time. The number of songs sang by Gumi, Ia, and Mayu must be $gumi$, ia, and $mayu$, respectively.

They soon realized that there are many ways of making the album. Two albums are considered different if there is a song that is sung by a different set of Vocaloids. Let X be the number of possible albums. Since the number X can be quite large, compute and print the number (X modulo 1,000,000,007). S will be between 1 and 50, inclusive. $gumi$, ia and $mayu$ will be each between 1 and S, inclusive.

For the first sample test case: In this case, there are 3 songs on the album. And Gumi, Ia, Mayu will each sing one song. There are 3*2*1 = 6 ways how to choose which Vocaloid sings which song. For the second sample test case: Gumi will sing all three songs. Ia and Mayu can each choose which one song they want to sing. For the third sample test case: It is not possible to record 50 songs if each Vocaloid can only sing 10 of them.

Input

The input will contain several test case, each in a line. The integer values S, $gumi$, ia, and $mayu$.

Output

Print a line per test case, the solution to the problem.

The output must be written to the standard output.

Input sample	**Output sample**
3 1 1 1	6
3 3 1 1	9
50 10 10 10	0
18 12 8 9	81451692
50 25 25 25	198591037
3 3 1 1	9
2 1 1 1	6

D - Cow

Nombre del archivo a enviar: `cow.cpp`, *or* `cow.java`

Farmer John had N cows numbered 0 to $N-1$. One day he saw K cows running away from his farm. Fox Brus computed the sum of the numbers of the escaped cows. She only told John that the sum was divisible by N.

Your task is to help John by counting the number of possible sets of escaped cows. This number may be very big, so printindex it modulo $1,000,000,007$. N will be between 1 and 1,000, inclusive. K will be between 1 and 47, inclusive. K will be less than or equal to N. For the first sample: 7 cows are numbered 0 to 6 and 4 of them run away. Possible sets of escaped cows are $\{0,1,2,4\}$, $\{0,3,5,6\}$, $\{1,2,5,6\}$, $\{1,3,4,6\}$, $\{2,3,4,5\}$.

Input

The input will contain several test case, each in a line. The integers N and K

Output

Print a line per test case, the solution to the problem.

The output must be written to the standard output.

Input sample	Output sample
7 4 1 1 58 4 502 7 1000 47 2 1	5 1 7322 704466492 219736903 1

E - Changer

Nombre del archivo a enviar: `changer.cpp`, *or* `changer.java`

You are converting old code for a new compiler version. Each $->$ string should be replaced by "." , but this replacement shouldn't be done inside comments. A comment is a string that starts with "//" and terminates at the end of the line.

You will be given a sequence of lines; a program, containing the old code. Printa a sequence of lines containing the converted version of the code.

Input

The input will consist in several test cases; each test case will consist of a line with an integer N between 1 and 50 and a program in the next N lines. You can assume the following conditions will always be true. Each element of program will contain between 1 and 50 characters, inclusive. Each character in program will have ASCII value between 32 and 127, inclusive.

Output

For each test case, print the converted program, the answer, follow the format on the output example.

The output must be written to the standard output.

Input sample	**Output sample**

4
Test* t = new Test();
t->a = 1;
t->b = 2;
t->go(); // a=1, b=2 --> a=2, b=3
3
---> // the arrow --->>
--- 	
> // the parted arrow
1
->-> // two successive arrows ->->
	Test* t = new Test();
	t.a = 1;
	t.b = 2;
	t.go(); // a=1, b=2 --> a=2, b=3
	--. // the arrow --->>
	--- 	
	> // the parted arrow
	.. // two successive arrows ->->

F - Vocabulary

Nombre del archivo a enviar: `vocabulary.cpp`, *or* `vocabulary.java`

Little Teddy and Little Tracy are now learning how to speak words. Their mother, of course, doesn't want them to speak bad words. According to her definition, a word W is bad if at least one of the following conditions hold (see the notes in input specification section for definitions):

W contains the string badPrefix as a prefix. W contains the string badSuffix as a suffix. W contains the string badSubstring as a contiguous substring that is neither a prefix nor a suffix of W. You are given a vocabulary representing the words that Teddy and Tracy are going to learn. Find and print the number of bad words in vocabulary.

Input

The input will consist in several test cases; each test case will consist of three lines, the first line will contain badPrefix, badSuffix and badSubstring, separated by single spaces and with out leading or trailing spaces, the second line of each test case will contain the vocabulary, separated by single spaces and with out leading or trailing spaces. You can assume the following conditions will always be true: A prefix of a string is obtained by removing zero or more contiguous characters from the end of the string. A suffix of a string is obtained by removing zero or more contiguous characters from the beginning of the string. badPrefix, badSuffix, and badSubstring will each contain between 1 and 50 characters, inclusive. vocabulary will contain between 1 and 50 elements, inclusive. Each element vocabulary will contain between 1 and 50 characters, inclusive. Each character of badPrefix, badSuffix, and badSubstring will be between 'a' and 'z', inclusive. Each character in vocabulary will be between 'a' and 'z', inclusive. All elements of vocabulary will be distinct.

Output

For each test case, print one line with the answer, follow the format on the output example.

The output must be written to the standard output.

Input sample	Output sample
bug bug bug	3
buggy debugger debug	3
a b c	0
a b tco	1
cut sore scar	3
scary oscar	
bar else foo	
foofoofoo foobar elsewhere	
pre s all	
all coders be prepared for the challenge phase	

G - Test

Nombre del archivo a enviar: `test.cpp`, *or* `test.java`

You are developing a new software calculator. During the testing phase of the software you have found that the test cases use different symbols as the decimal point of floating numbers. Moreover some test cases contain useless space symbols. Now you want to bring the numbers to a unified format.

You will be given a sequence of numbers. Remove all space symbols (ASCII code 32) from the given numbers and replace each non-digit symbol with a dot symbol ('.'). You should not make any other changes to the numbers.

Input

The input will consist in several test cases; each test case will consist of a line with a integer N indicating the number of lines of numbers. Then follow N lines, each line with a string "number" of que given sequence numbers. You can assume the following conditions will always be true numbers will have between 1 and 50 elements, inclusive. Each element of numbers will contain between 1 and 50 characters, inclusive. Each character in numbers will have ASCII code between 32 and 127, inclusive. Each element of numbers will contain at most one non-space non-digit symbol. Each element of numbers will contain at least one digit.

Output

For each test case, print one line with the answer, follow the format on the output example.

The output must be written to the standard output.

Input sample	Output sample
3 1.5 2$ 3 12 3 6 ,5 3, .5 3. 000,000 000 000 3 263C45233 2364A56 B273664	1.5 2.3 123 .5 3. .5 3. 000.000 000000 263.45233 2364.56 .273664

H - Dancing

Nombre del archivo a enviar: `dancing.cpp`, *or* `dancing.java`

A sentence is called dancing if its first letter is uppercase and the case of each subsequent letter is the opposite of the previous letter. Spaces should be ignored when determining the case of a letter. For example, "A b Cd" is a dancing sentence because the first letter ('A') is uppercase, the next letter ('b') is lowercase, the next letter ('C') is uppercase, and the next letter ('d') is lowercase.

You will be given a sentence. Turn the sentence into a dancing sentence by changing the cases of the letters where necessary. All spaces in the original sentence must be preserved.

Input

The input will consist in several test cases; each test case will consist of a line with the given sentence. You can assume the following conditions will always be true: The given sentence will contain between 1 and 50 characters, inclusive. Each character in the given sentence will be a letter ('A'-'Z','a'-'z') or a space (' '). The given sentence will contain at least one letter ('A'-'Z','a'-'z').

Output

For each test case, print one line with the answer, follow the format on the output example.

The output must be written to the standard output.

Input sample	**Output sample**

`This is a dancing sentence`
` This is a dancing sentence`
`aaaaaaaaaaa`
`z`
	`ThIs Is A dAnCiNg SeNtEnCe`
	` ThIs Is A dAnCiNg SeNtEnCe`
	`AaAaAaAaAaA`
	`Z`

I - Machine

Nombre del archivo a enviar: `machine.cpp`, *or* `machine.java`

We have a String, let's call it originalWord. Each character of originalWord is either 'a' or 'b'. Timmy claims that he can convert it to finalWord using exactly k moves. In each move, he can either change a single 'a' to a 'b', or change a single 'b' to an 'a'.

You are given the Strings originalWord and finalWord, and the integer value k. Determine whether Timmy may be telling the truth. If there is a possible sequence of exactly k moves that will turn originalWord into finalWord, print "POSSIBLE" (quotes for clarity). Otherwise, print "IMPOSSIBLE".

Input

The input will consist in several test cases; each test case will consist of a line with originalWord, finalWord and k, each token will be separated by a single space and there will not be leading or trailing spaces. You can assume the following conditions will always be true: Timmy may change the same letter multiple times. Each time counts as a different move. originalWord will contain between 1 and 50 characters, inclusive. finalWord and originalWord will contain the same number of characters. Each character in originalWord and finalWord will be 'a' or 'b'. k will be between 1 and 100, inclusive.

Output

For each test case, print one line with the answer, follow the format on the output example.

The output must be written to the standard output.

Input sample	Output sample
aababba bbbbbbb 2	IMPOSSIBLE
aabb aabb 1	IMPOSSIBLE
aaaaabaa bbbbbabb 8	POSSIBLE
aaa bab 4	POSSIBLE
aababbabaa abbbbaabab 9	IMPOSSIBLE

J - Power

Nombre del archivo a enviar: `power.cpp`, *or* `power.java`

A number n taken to the falling factorial power k is defined as n*(n-1)*...*(n-k+1). We will denote it by n+++k. For example, 7+++3=7*6*5=210. By definition, n+++1=n.

We will now continue this definition to the non-positive values of k using the following fact: (n-k)*(n+++k)=n+++(k+1), or, in other words, n+++k=(n+++(k+1))/(n-k). It is directly derived from the above definition.

By using it, we find:

n+++0=n+++1/(n-0)=1,
n+++(-1)=n+++0/(n+1)=1/(n+1),
n+++(-2)=1/(n+1)/(n+2),
and, in general, n+++(-k)=1/(n+1)/(n+2)/.../(n+k).
For example, 3+++(-1)=1/4=0.25, 2+++(-3)=1/3/4/5=1/60=0.016666...
Given a positive integer n $(1 <= n <= 10)$ and an integer k $(-5 <= n <= 5)$, find and print a real number containing the value of n taken to the falling factorial power of k.

Input

The input will consist in several test cases; each test case will consist of a line with n and k.

Output

For each test case, print one line with the answer, follow the format on the output example. The answer must be rounded and printed with sex decimal positions.

The output must be written to the standard output.

Input sample	Output sample
7 3	210.000000
10 1	10.000000
5 0	1.000000
3 -1	0.250000
2 -3	0.016667

K - Chess

Nombre del archivo a enviar: `chess.cpp`, *or* `chess.java`

You have decided that too many people do not know how to play chess. So, in an effort to teach the rules you must write some software that helps to understand how chess-pieces affect one another. Your current project involves the knight and its ability to threaten one or more pieces at once. The knight has an unusual style of "jumping" around the board. One move consists of traveling two squares in one of the four cardinal directions, followed by one square perpendicular to the original direction. For example, if a knight is on (0,0), it may move to (2,1), (2,-1), (1,2), (1,-2), (-2, 1), (-2,-1), (-1,2), or (-1,-2). In addition, if a piece is on any of those locations, it is threatened by the knight on (0,0). You will be given a sequence of pieces; pieces, where each element is a comma delimited set of coordinates. Every element in pieces is formatted as "¡x-coordinate¿,¡y-coordinate¿" (quotes and angle brackets for clarity). Calculate and print a sequence of lines where each line represents a position from which a knight threatens every piece in pieces. Your printed sequence of lines must be in the same format as pieces and sorted in increasing order by the x-coordinate. If two sets of coordinates have the same x-coordinate, the one with the smaller y-coordinate must come first.

Input

The input will consist in several test cases; each test case will consist of a line an Integer N, indicating the numbers of lines that follow, each of the next N lines will be an element of pieces. You can assume the following conditions will always be true: pieces will contain between 1 and 8 elements, inclusive. Each element in pieces will be formatted as "¡x-coordinate¿,¡y-coordinate¿" (quotes and angle brackets for clarity). Each ¡x-coordinate¿ will be an integer between -10000 and 10000, inclusive and will not contain leading zeros. Each ¡y-coordinate¿ will be an integer between -10000 and 10000, inclusive and will not contain leading zeros. Each element in pieces will be unique.

Output

For each test case, print several lines with the answer; follow the format on the output example.

The output must be written to the standard output.

Input sample	Output sample
1 0,0 2 2,1 -1,-2 2 0,0 2,1 3 -1000,1000 -999,999 -999,997	8 found -2,-1 -2,1 -1,-2 -1,2 1,-2 1,2 2,-1 2,1 2 found 0,0 1,-1 0 found 1 found -1001,998 8 found -10002,-10001 -10002,-9999 -10001,-10002 -10001,-9998 -9999,-10002 -9999,-9998 -9998,-10001 -9998,-9999

L - Haar

Nombre del archivo a enviar: `haar.cpp`, *or* `haar.java`

The Haar wavelet transform is possibly the earliest wavelet transform, introduced by Haar in 1909. The 1-dimensional version of this transform operates on a sequence of integer data as follows: First separate the sequence into pairs of adjacent values, starting with the first and second values, then the third and fourth values, etc. Next, calculate the sums of each of these pairs, and place the sums in order into a new sequence. Then, calculate the differences of each of the pairs (subtract the second value of each pair from the first value), and append the differences in order to the end of the new sequence. The resulting sequence is a level-1 transform. Note that this requires the input sequence to have an even number of elements.

The above describes a level-1 transform. To perform a level-2 transform, we repeat the above procedure on the first half of the sequence produced by the level-1 transform. The second half of the sequence remains unchanged from the previous level. This pattern continues for higher level transforms (i.e., a level-3 transform operates with the first quarter of the sequence, and so on). Note that this is always possible when the number of elements is a power of 2.

Given a sequence of integers; data and an integer L. Find and print the level-L Haar transform of the data.

Input

The input will consist in several test cases; each test case will consist of two lines, a line with the data, and a line with L. You can assume the following conditions will always be true The given data will contain exactly 2, 4, 8, 16 or 32 elements. Each element of data will be between 0 and 100 inclusive. L will be between 1 and log_2(Number of elements in data) inclusive.

Output

For each test case, print one line with the answer, follow the format on the output example.

The output must be written to the standard output.

Input sample	Output sample
1 2 3 5 1 1 2 3 5 2 1 2 3 4 4 3 2 1 3 94 47 46 28 39 89 75 4 28 62 69 89 34 55 81 24 2	3 8 -1 -2 11 -5 -1 -2 20 0 -4 4 -1 -1 1 1 215 207 248 194 67 49 -68 -16 47 18 -

M - Sequence

Nombre del archivo a enviar: `sequence.cpp`, *or* `sequence.java`

Little Rudolph had an important sequence of positive integers. The sequence consisted of N positive integers a_0, a_1, .., a_{N-1}.

Rudolph wrote the sequence onto the blackboard in the classroom. While Rudolph had gone out, little Arthur came into the classroom and saw the sequence. Arthur likes to play with numbers as much as he likes to give his friends puzzles. So he did the following: First, he wrote a '+' or a '-' between each pair of consecutive numbers (possibly using different signs for different pairs of numbers). Next, for each sign he computed the result of the corresponding operation and wrote it under the sign. I.e., if he used the '+' sign between a_i and a_{i+1}, he would write the sum a_i+a_{i+1} under this '+' sign. Similarly, if he used the '-' sign between a_i and a_{i+1}, he would write the difference a_i-a_{i+1}. In this way he obtained a new sequence of $N-1$ numbers b_0, b_1, .., b_{N-2}. Finally, he erased the original sequence. Now there was only the operator sequence o_0, o_1, .., o_{N-2} and the resulting number sequence b_0, b_2, .., b_{N-2} left on the blackboard. For example, if the original sequence was $\{1, 2, 3, 4\}$, and Arthur wrote operators +, -, +, then the content of the blackboard changed like this:

```
1   2   3   4    ->    1 + 2 - 3 + 4    ->    1 + 2 - 3 + 4    ->     +  -  +
                                                 3  -1   7             3 -1  7
```

When Rudolph returned, he was shocked as his important sequence had disappeared. Arthur quickly told him what operations he had performed and that Rudolph has to simply reconstruct the original sequence.

Unfortunately, little Arthur did not realize that it is not necessarily possible to determine the original sequence uniquely. For example, both original sequences $\{1, 2, 3, 4\}$ and $\{2, 1, 2, 5\}$ lead to the same sequence $\{3, -1, 7\}$ when operator sequence is $\{+, -, +\}$.

The only thing Rudolph remembers about his original sequence is that all the integers were positive. Rudolph now wants to count all sequences of positive integers that match the blackboard. You are given an integer sequence called B and line of operators called operators both containing N-1 elements. The i-th element of B is the number bi and i-th element of operators will be '+' or '-', meaning that the i-th operator is + or -, respectively. Find and print the number of different positive integer sequences A that lead to sequence B when operators operators are used in the way described. If there are infinitely many such sequences, just print -1. Note that there may be test cases where no valid sequence A exists. For such test cases the correct value to print is 0.

Input

The input will consist in several test cases; each test case will consist of two lines, a line with the sequence B, and a line with the operators. You can assume the following conditions will always be true: It is guaranteed that the correct answer will always fit into the 32-bit signed integer type. The integer 0 (zero) is not positive. It may not occur in Rudolph's original sequence. B will contain between 1 and 50 elements, inclusive. operators will contain the same number of characters as the

number of elements in B. Each element of B will be between -1000000000 (-10^9) and 1000000000 (10^9), inclusive. Each character in operators will be either '+' or '-' (quotes for clarity).

Output

For each test case, print one line with the answer, follow the format on the output example.

The output must be written to the standard output.

Input sample	**Output sample**
3 -1 7 +-+ 1 - 1 + 10 + 540 2012 540 2012 540 2012 540 -+-+-+-	2 -1 0 9 1471

N - Jaguars

Nombre del archivo a enviar: `jaguars.cpp`, *or* `jaguars.java`

There are N animals numbered 0 to N-1 in a zoo. Each animal is a jaguar or a cat. Their heights are pairwise distinct.

Fox Jiro can't distinguish between jaguars and cats, so he asked the following question to each animal: "How many animals of the same kind as you are taller than you?" Each jaguar tells the number of jaguars taller than him, and each cat tells the number of cats taller than her. The differences of heights are slight, so Fox Jiro can't tell which animals are taller than other animals. However, each animal is able to determine which animals are taller that him and which ones are shorter.

The answer given by the i-th animal is answersi. Given these numbers, find and print the number of configurations resulting in exactly those numbers, assuming everyone tells the truth. Two configurations are different if there exists an i such that the i-th animal is a jaguar in one configuration and cat in the other configuration.

Input

The input will consist in several test cases; each test case will consist of a line with the sequence answer. You can assume the following conditions will always be true: answers will contain between 1 and 40 elements, inclusive. Each element of answers will be between 0 and 40, inclusive.

Output

For each test case, print one line with the answer, follow the format on the output example.

The output must be written to the standard output.

Input sample	**Output sample**
0 1 2 3 4 5 8 0 0 0 0 0 0 1 0 2 0 1 1 0 1	2 0 0 8 0

O - Kingdom

Nombre del archivo a enviar: `kingdom.cpp`, *or* `kingdom.java`

King Dengklek once planted N trees, conveniently numbered 0 through N-1, along the main highway in the Kingdom of Ducks. As time passed, the trees grew beautifully. Now, the height of the i-th tree is heights[i] units.

King Dengklek now thinks that the highway would be even more beautiful if the tree heights were in strictly ascending order. More specifically, in the desired configuration the height of tree i must be strictly smaller than the height of tree i+1, for all possible i. To accomplish this, King Dengklek will cast his magic spell. If he casts magic spell of level X, he can increase or decrease the height of each tree by at most X units. He cannot decrease the height of a tree into below 1 unit. Also, the new height of each tree in units must again be an integer.

Of course, a magic spell of a high level consumes a lot of energy. Find and print the smallest possible non-negative integer X such that King Dengklek can achieve his goal by casting his magic spell of level X.

Input

The input will consist in several test cases; each test case will consist of a line with the given sequence of heights. You can assume the following conditions will always be true: heights will contain between 2 and 50 elements, inclusive. Each elements of heights will be between 1 and 1,000,000,000, inclusive.

Output

For each test case, print one line with the answer, follow the format on the output example.

The output must be written to the standard output.

Input sample	**Output sample**

`9 5 11`
`5 8`
`1 1 1 1 1`
`548 47 58 250 2012`
	`3`
	`0`
	`4`
	`251`

P - Sea

Nombre del archivo a enviar: `sea.cpp`, *or* `sea.java`

Bob's father bought him a toy map of islands and seas. The map is a two-dimensional grid where each cell is either 'x' or '.'. A sea is defined as a maximal connected group of '.' cells, where two '.' cells are connected if they are vertically or horizontally adjacent. An island is defined as a maximal connected group of 'x' cells, where two 'x' cells are connected if they are vertically, horizontally, or diagonally adjacent. An island has a level of 0 if it contains no other islands. An island has a level of K+1 if it contains one or more islands and the highest level of a contained island is K. An island A contains island B if A and B are different and, if you start sailing from any point of island B, you won't be able to sail out of island A (you can sail only horizontally and vertically, but not diagonally).

For example, the given map below has 5 islands with level 0 (islands 0 - 4 on the right picture) and one island with level 1 (island 5). Please note that starting at island 3, you can not sail outside island 5 (you can not sail diagonally), but its possible get out of island 1 when starting at island 4.

```
xxx.x...xxxxx        000.0...11111
xxxx....x...x        0000....1...1
........x.x.x        ........1.4.1
..xxxxx.x...x        ..55555.1...1
..x...x.xxx.x        ..5...5.111.1
..x.x.x...x..        ..5.3.5...1..
..x...x...xxx        ..5...5...111
...xxxxxx....        ...555555....
x............        2............
```

Given seaMap, find and print several integers (may be one), where the k-th element is the number of islands of level k. The output to the problem must contain exactly (m + 1) elements, where m is the highest level of an island in the map.

Input

The input will consist in several test cases; each test case will consist of a line with R and C,between 1 and 50, the numbers of rows and cols of the given seaMap, the next R lines will contain the given seaMap. You can assume the following conditions will always be true: seaMap will contain between 1 and 50 elements, inclusive. Each element of seaMap will contain between 1 and 50 characters, inclusive. Each element of seaMap will contain the same number of characters. Each element of seaMap will contain only '.' and lowercase 'x' characters.

Output

For each test case, print one line with the answer, follow the format on the output example.

The output must be written to the standard output.

Input sample	Output sample
1 1 x 5 5 xxxxx x...x x.x.x x...x xxxxx 10 5 xxxxx x...x x.x.x x...x xxxxx xxxxx x...x x.x.x x...x xxxxx 2 2 .. .. 6 12 ....xxxx. ..xxx.x...x. ..x..x..x.x. ..x.x.x...x. ..xx...xxx..	{1} {1, 1} {2, 1} {}