Problem A. Walking around Berhattan

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	64 megabytes

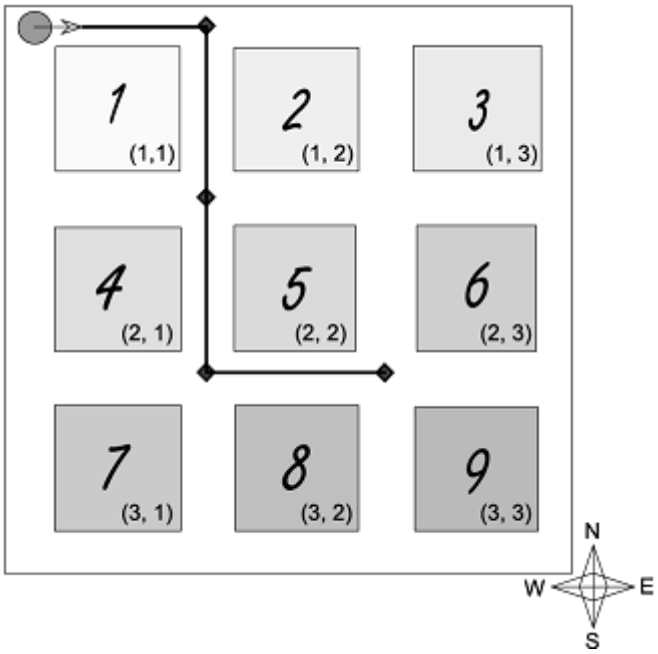
As you probably know, Berhattan is a district of Berland's largest city and it consists of equal square blocks. There are n block lines in the east-west direction and m block lines in the south-north direction. The map shows Berhattan as a rectangle with n rows and m columns, so there are $n \times m$ blocks in total.

There are $n + 1$ streets running parallel in the east-west direction (horizontally), and there are $m + 1$ avenues running parallel in the south-north direction (vertically). Streets and avenues split the district into blocks and separate Berhattan from other districts of Berland. Each block in Berhattan is characterized by its *beauty* b_{ij}.

A pedestrian can walk only along streets and avenues. When the pedestrian walks along any of four sides of a block, we say he passes the block. Every time the pedestrian passes a block his *satisfaction* is increased by b_{ij}. If the pedestrian has already passed the block one or more times his satisfaction is increased only by $b_{ij}/2$ rounded down when he passes the block again.

You are given the map of Berhattan with the information about the blocks' beauty and the pedestrian's path along the streets and avenues. The path is given as a string containing letters '`L`', '`R`' and '`M`', where '`L`' means a 90 degree left turn, '`R`' means a 90 degree right turn, and '`M`' means walking one block forward by a street or avenue. Facing the east, the pedestrian starts his path in the north-west corner of Berhattan having zero satisfaction level. His path can cross itself and go along the same streets or avenues several times. Pedestrian's satisfaction is increased every time he moves according to the rules described above.

Your task is to calculate the total satisfaction the pedestrian will get after finishing his route.



Picture of the sample test

Input

The first line of input contains two integers n and m ($1 \le n, m \le 100$), where n is a number of block lines in Berhattan running in the east-west direction, and m is a number of block lines in Berhattan running in the south-north direction. The following n lines contain m digits each. The j-th digit of the i-th line

represents b_{ij} ($0 \le b_{ij} \le 9$) — the beauty of the corresponding block. The last line of input contains a path in the format specified above. The path consists of 1 up to 500 characters, inclusively. It is guaranteed that the given path doesn't go outside Berhattan.

Output

Print a single integer to the output — the total pedestrian's satisfaction.

Examples

input.txt	output.txt
3 3 123 456 789 MRMMLM	22

Problem B. Kakuro

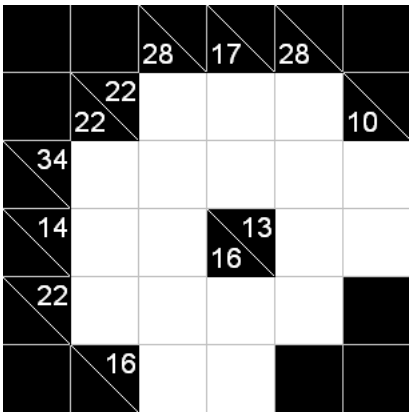
Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	64 megabytes

Kakuro puzzle is played on a $n \times m$ grid of «black» and «white» cells. Apart from the top row and leftmost column which are entirely black, the grid has some amount of white cells which form «runs» and some amount of black cells. «Run» is a vertical or horizontal maximal one-lined block of adjacent white cells. Each row and column of the puzzle can contain more than one «run». Every white cell belongs to exactly two runs — one horizontal and one vertical run. Each horizontal «run» always has a number in the black half-cell to its immediate left, and each vertical «run» always has a number in the black half-cell immediately above it. These numbers are located in «black» cells and are called «clues».

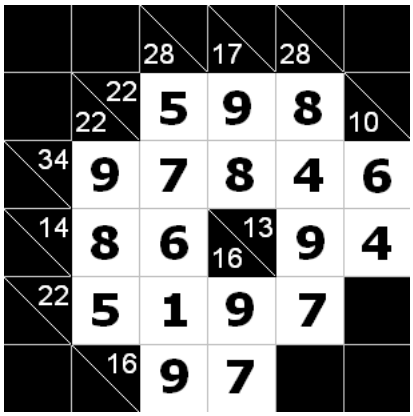
The rules of the puzzle are simple:

- place a single digit from 1 to 9 in each «white» cell
- each digit may only be used once in each «run»
- for all runs, the sum of all digits in a «run» must match the clue associated with the «run»

Given the grid, your task is to find a solution for the puzzle.



Picture of the first sample input



Picture of the first sample output

Input

The first line of input contains two integers n and m ($2 \le n, m \le 6$) — the number of rows and columns correspondingly. Each of the next n lines contains descriptions of m cells. Each cell description is one of the following 5-character strings:

- `.....` — «white» cell;
- `XXXXX` — «black» cell with no clues;
- `AA\BB` — «black» cell with one or two clues. `AA` is either a 2-digit clue for the corresponding vertical run, or «`XX`» if there is no associated vertical run. `BB` is either a 2-digit clue for the corresponding horizontal run, or «`XX`» if there is no associated horizontal run.

The first row and the first column of the grid will never have any white cells. The given grid will have at least one «white» cell.

It is guaranteed that the given puzzle has at least one solution.

Output

Print n lines to the output with m cells in each line. For every «black» cell print '_' (underscore), for every «white» cell print the corresponding digit from the solution. Delimit cells with a single space, so that each row consists of $2m - 1$ characters.

If there are many solutions, you may output any of them.

Examples

input.txt

```
6 6
XXXXX XXXXX 28\XX 17\XX 28\XX XXXXX
XXXXX 22\22 ..... ..... ..... 10\XX
XX\34 ..... ..... ..... ..... .....
XX\14 ..... ..... 16\13 ..... .....
XX\22 ..... ..... ..... ..... XXXXX
XXXXX XX\16 ..... ..... XXXXX XXXXX
```

output.txt

```
_ _ _ _ _ _
_ _ 5 9 8 _
_ 9 7 8 4 6
_ 8 6 _ 9 4
_ 5 1 9 7 _
_ _ 9 7 _ _
```

input.txt

```
3 3
XXXXX 04\XX XXXXX
XX\04 ..... XXXXX
XXXXX XXXXX XXXXX
```

output.txt

```
_ _ _
_ 4 _
_ _ _
```

Problem C. Electrician

Input file:	input.txt
Output file:	output.txt
Time limit:	1 second
Memory limit:	64 megabytes

An electrician Vasya has got an assignment to solder n wires. His boss specified the requirements precisely, so for each wire Vasya knows exactly where its endpoints should be soldered to. Two identifiers a_i, b_i are given for each wire, meaning that one endpoint of the wire should be soldered to the place a_i, and the other endpoint should be soldered to the place b_i. It doesn't matter which endpoint will be soldered to which place. Also each wire has two more characteristics r_i and p_i, where r_i is its reliability and p_i is its cost.

The only way for Vasya to express himself in such a rigorous constraints is to choose an order, and solder all wires in this order, one after another. As an experienced electrician Vasya knows what a short circuit is — it occurs when a scheme contains a cycle, in other words when there is more than one simple path over wires from one place to another. So, if a short circuit occurs after a wire is soldered, the least reliable wire in the cycle burns out (you may think that the least reliable wire disappears from the scheme). If there are several least reliable wires in the cycle, the one of them which was soldered earlier burns out. It is clear that after a wire burns out, the scheme doesn't have any cycles.

When Vasya is done with soldering, he ends up with a scheme of soldered wires. So, he wants to solder all wires in such an order, that the total cost of wires in a resulting scheme will be as maximal as possible.

Input

The first line of input contains a single integer n ($1 \le n \le 30000$). Each of the following n lines contains four integer numbers a_i, b_i, r_i, p_i ($1 \le a_i, b_i, r_i, p_i \le 10^9; a_i \ne b_i$), where a_i and b_i are identifiers of the places for endpoints of i-th wire, r_i is the reliability of the wire, and p_i is the cost of the wire. There can be more than one wire between any pair of places.

Output

Print the required maximal total cost to the first line of output. Print the order of wires for soldering to the second line, delimiting wire indices with a single space.

You may print any solution if there are many of them.

Examples

input.txt	output.txt
4 10 20 5 3 20 11 5 2 10 11 7 1 1 2 1 1	5 2 3 1 4

Note

In the sample test Vasya can choose any order with the only rule: the second wire should be soldered before the first one. If he violates the rule, the total cost will be 4 instead of 5.

Problem D. Sequence analysis

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	10 seconds
Memory limit:	64 megabytes

You are given a sequence of signed 64-bit integers defined as follows:

- $x_0 = 1$,
- $x_{i+1} = (A \cdot x_i + x_i \mod B) \mod C$,

where `mod` is a remainder operator. All arithmetic operations are evaluated without overflow checking. Use standard "remainder" operator for programming languages (it differs from the mathematical version; for example $-5 \mod 2 = -1$ in programming, while $-5 \mod 2 = 1$ in mathematics). Use "`long long`" type in C++, "`long`" in Java and "`int64`" in Delphi to store x_i and all other values.

Let's call a sequence element x_p *repeatable* if it occurs later in the sequence — meaning that there exists such q, $q > p$, that $x_q = x_p$. The *first repeatable* element M of the sequence is such an element x_m that x_m is *repeatable*, and none of the x_p where $p < m$ are *repeatable*.

Given A, B and C, your task is to find the index of the second occurrence of the *first repeatable* element M in the sequence if the index is less or equal to $2 \cdot 10^7$. Per definition, the first element of the sequence has index 0.

Input

The only line of input contains three signed 64-bit integers: A, B and C ($B > 0, C > 0$).

Output

Print a single integer — the index of the second occurence of the *first repeatable* member if it is less or equal to $2 \cdot 10^7$. Print -1 if the index is more than $2 \cdot 10^7$.

Examples

input.txt
2 2 9
output.txt
4

input.txt
2305843009213693951 1 9223372036854775807
output.txt
5

input.txt
-2 1 5
output.txt
4

Note

In the first sample test the sequence starts with the following numbers: 1, 3, 7, 6, 3, 7. The *first repeatable* element is 3. The second occurence of 3 has index 4.

In the second sample test the sequence starts with the following numbers: 1, 2305843009213693951, -4611686018427387903, 6917529027641081855, 0, 0, 0. The *first repeatable* element is 0. The second occurence of 0 has index 5.

In the third sample test the sequence starts with the following numbers: 1, -2, 4, -3, 1, -2, 4. The *first repeatable* element is 1. The second occurence of 1 has index 4.

Problem E. Meetings

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	64 megabytes

Two cities A and B are connected by a straight road that is exactly l meters long. At the initial moment of time a cyclist starts moving from city A to city B at a speed v_1 meters/second, and a pedestrian starts moving from city B to city A at a speed v_2 meters/second. When one of them reaches a city, the road ends, so the person has to turn around and start moving in the opposite direction by the same road, keeping the original speed. As a result, the cyclist and the pedestrian are traveling between cities A and B indefinitely.

Your task is to calculate the number of times they will meet during the first t seconds. If they meet in exactly t seconds after the initial moment of time, this meeting should also be counted.

Input

The only line of input contains four integer numbers: l, v_1, v_2 and t. All numbers are between 1 and 10^9, inclusively.

Output

Print a single integer — the number of times the cyclist and the pedestrian will meet during the first t seconds.

Examples

input.txt	output.txt
1000 10 1 200	2
4 4 3 4	4

Problem F. The Monochrome Picture

Input file:	input.txt
Output file:	output.txt
Time limit:	1 second
Memory limit:	64 megabytes

An artist Kalevich is very ambitious and he has many different achievements over the years of his work. Kalevich became extremely famous when he first produced the largest digital picture in the world, setting a new world record in digital painting. It was a great victory with a very unusual image — a billion pixels in width, and... only one pixel in height. The win changed the entire Kalevich's life, so starting from that memorable moment all his digital masterpieces have the height of 1 pixel.

Recently Kalevich was invited to an exhibition in order to demonstrate the best picture he has ever painted. The picture is n pixels in width, 1 pixel in height, and it is called "The Monochrome Snake". As you have already guessed, the painting is indeed monochrome, so the i-th pixel is characterized by a single integer c_i from 0 to 10^6 that is a grayscale representation of its color.

Many visitors at the exhibition have never seen any pictures with colors different from the standard 24-bit RGB, so they look at Kalevich's masterpiece with a great suspicion. Kalevich realized that the visitors do not like monochrome pictures at all, and what is even worse, if the colors of two adjacent pixels in a monochrome picture differ exactly by one, the visitors get angry and go away. Kalevich feels really nervous about this, so he wants to improve his painting in order to please the exigent visitors and keep them at the exhibition. At the same time he wants to preserve the idea of the picture — the snake should be still recognizable, so the only change he wants to make is to delete some pixels here and there. When he deletes a pixel, the width of the painting decreases by 1 of course. Kalevich will be satisfied with the result if $|r_i - r_{i+1}| \neq 1$ for all $i = 1 \ldots m - 1$, where r is the final masterpiece and m is its length.

Your task is to help Kalevich and write a program that will help him to delete the minimum number of pixels from the picture, so that the resulting masterpiece does not have any two adjacent pixels with the colors that differ exactly by one.

Input

The first line of input contains a single integer n ($1 \le n \le 10^5$). The second line of input contains n integers separated by spaces — pixel colors $c_1, c_2, \ldots, c_n$ ($0 \le c_i \le 10^6$).

Output

To the first line of output print the minimum number of pixel deletions t that are needed to satisfy Kalevich's requirements. To the second line print m integer numbers ($m = n - t$) — the masterpiece that is left after t pixel deletions.

If there are many solutions, you may output any of them.

Examples

input.txt	output.txt
6 4 2 2 1 1 1	2 4 1 1 1
5 1 2 3 2 1	2 1 3 1

Problem G. Plural Form of Nouns

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	64 megabytes

In the English language, nouns are inflected by grammatical number — that is singular or plural. In this problem we use a simple model of constructing plural from a singular form. This model doesn't always make English plural forms correctly, but it works in most cases. Forget about the real rules you know while solving the problem and use the statement as a formal document.

You are given several nouns in a singular form and your program should translate them into plural form using the following rules:

- If a singular noun ends with *ch*, *x*, *s*, *o* the plural is formed by adding *es*. For example, *witch* → *witches*, *tomato* → *tomatoes*.
- If a singular noun ends with *f* or *fe*, the plural form ends with *ves*. For example, *leaf* → *leaves*, *knife* → *knives*. Pay attention to the letter *f* becoming *v*.
- Nouns ending with *y* change the ending to *ies* in plural. For example, *family* → *families*.
- In all other cases plural is formed by adding *s*. For example, *book* → *books*.

Input

The first line of input contains a single positive integer n ($1 \le n \le 10$) — the number of words to be processed. The following n lines contain one word each. A word consists from 2 to 25 lowercase Latin letters. It is not guaranteed that the given words are real English words from vocabulary.

Output

Print n given words in their plural forms on separate lines. Keep the words in the same order as they are given in the input.

Examples

input.txt	output.txt
3 contest hero lady	contests heroes ladies

Problem H. Annuity Payment Scheme

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	64 megabytes

At the peak of the Global Economic Crisis BerBank offered an unprecedented credit program. The offering was so attractive that Vitaly decided to try it. He took a loan of s burles for m months with the interest rate of p percent.

Vitaly has to follow the scheme of annuity payments, meaning that he should make fixed monthly payments — x burles per month. Obviously, at the end of the period he will pay $m \cdot x$ burles to the bank in total.

Each of the monthly payments is divided by BerBank into two parts as follows:

- The first part a_i is used to pay off the percent p of the current debt. It's clear that $a_i = s' \cdot p/100$ where $s' = s$ for the first month and equals to the remaining debt for each of the subsequent months.
- The second part b_i is used to pay off the current debt. The sum of all b_i over the payment period is equal to s, meaning that the borrower needs to pay off the debt completely by decreasing it from s to 0 in m months.

BerBank uses calculations with floating-point numbers, and the value of x is uniquely determined by s, m and p.

For example, if $s = 100, m = 2, p = 50$ then $x = 90$. For the first month $a_1 = s' \cdot p/100 = s \cdot p/100 = 50$ and $b_1 = 90 - 50 = 40$. For the second month $a_2 = (100 - 40) \cdot 50/100 = 30$, so $b_2 = 90 - 30 = 60$ and the debt is paid off completely.

Your task is to help Vitaly and write a program that computes x given the values of s, m and p.

Input

The single line of the input contains three integers s, m and p ($1 \le s \le 10^6$, $1 \le m \le 120$, $0 \le p \le 100$).

Output

Output the single value of monthly payment x in burles. An absolute error of up to 10^{-5} is allowed.

Examples

input.txt	output.txt
100 2 50	90.00000

Problem I. Snow in Berland

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	64 megabytes

Winters are very snowy in Berland, and the current winter is not an exception. Each winter Berland government decides how to clean the roads in the country. The problem is particularly acute in the capital.

You may assume that the capital of Berland consists of n junctions and m one-way roads. Each road has two distinct junctions x_i, y_i as its end-points, and the traffic goes from x_i to y_i. There are w_i tons of snow on i-th road.

The government hired a private company "Snow White" to clean the city from the snow. Every day the company sends one truck for cleaning the roads — the truck starts from junction A, passes some route to junction B and stops. Single route can contain any road several times, and can pass through any junction (including A and B) several times.

So, the truck makes only one trip from junction A to junction B per day, and the truck's driver, of course, may not violate the traffic direction on the roads. The truck removes one ton of snow from each road it passes. If it passes the road several times during the same day, each time one ton of snow is removed from the road. Because capital residents may decide that the government spends the budget for nothing, the truck can not pass the road if there is no snow on it.

Some roads in the city have historical value due to the presence of government buildings, so this set of roads must be completely cleaned from snow. In other words each road from the specified set should not have snow after "Snow White"'s work. It's known that junction A is situated in the historical center of the capital, meaning that it is possible to reach any historical road from A, walking only along historical roads in the direction of their orientation or in the opposite direction. The direction of roads is not taken into account in this particular case, because we are talking about walking, not driving.

The government pays "Snow White" for each day of work, so "Snow White"'s top managers are looking for a way to work as many days as possible.

Your task is to find the sequence of routes from A to B which doesn't violate the rules described above. This sequence must completely clean all historical roads from snow. Obviously, the sequence should contain as many routes as possible.

Input

The first line of the input contains integer numbers n, m, A, B ($2 \le n \le 100; 0 \le m \le 5000; 1 \le A, B \le n; A \ne B$), where n — the number of junctions in the capital and m — the number of roads in it. The following m lines describe one-way roads, one road per line. Each line contains four integers x_i, y_i, w_i, t_i ($1 \le x_i, y_i \le n; x_i \ne y_i; 0 \le w_i \le 100; 0 \le t_i \le 1$), where x_i, y_i are the endpoints of the road, w_i — the amount of snow in tons on the road, and t_i — type of the road (0 means regular road, and 1 means historical road). There will be no more than one road between two junctions in each direction. It is possible to reach any historical road from A by walking along other historical roads (again, not taking into account the direction while walking)

Output

Write p — the maximum number of days "Snow White" can work. The next p lines should contain the chosen routes. Each route should be printed as a list of junctions that starts with A and ends with B, and all junction numbers should be separated by spaces. You may print routes in any order.

If there are many solutions, you may output any of them. If there is no solution, write a single integer 0 to the output.

Examples

input.txt	output.txt
4 7 1 4 1 2 3 1 2 1 100 0 2 4 1 0 1 3 1 0 3 4 4 0 2 3 2 1 1 4 2 0	6 1 3 4 1 4 1 4 1 2 4 1 2 3 4 1 2 3 4
3 3 1 2 1 3 2 0 3 2 3 0 1 2 1 0	3 1 3 2 1 3 2 1 2

Problem J. Choreographer Problem

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
Memory limit:	64 megabytes

As you probably know, choreography is the art of making dances. But it is not a so well-known fact that it is also the science of making dances.

There are n dancers numbered from 1 to n in the dance troupe, and all of them are working hard to create a new original dance. Before the dance starts each dancer puts on his special hand-made costume, and he has to wear this costume for the duration of the entire dance.

It is known that the costume of i-th dancer is similar to the costumes of $(i-1)$-th and $(i+1)$-th dancers (therefore the dancers are similar too), but at the same time, i-th dancer is not similar to any other dancer. Therefore, $(i-1)$-th and $(i+1)$-th dancers are not similar. If $n > 1$, then the first dancer has the only similar dancer, the last dancer has the only similar dancer, and all other dancers have exactly two similar dancers.

The dance starts and ends with an empty stage. The stage should not be empty during the dance. Each minute one of the following changes happens on the stage:

- one dancer (who is currently not on the stage) appears;
- one dancer (who is currently on the stage) leaves the stage;
- one dancer takes the place of another dancer similar to him/her (basically the dancers switch over — one dancer leaves the stage, while a similar dancer appears on the stage)

At every moment of time the stage must not have more than k dancers, because spectators may lose attention if there are too many dancers on the stage.

Now choreographer is thinking about arranging the dance in such a way that each set of the dancers containing no more than k dancers appears on the stage exactly once. Your job is to write a program to help the choreographer.

Input

The first line contains two positive integer numbers n, k $(1 \le n \le 20; 1 \le k \le n)$.

Output

Print the only line describing the dance. The substring «+i» means that the i-th dancer appears on the stage, the substring «-i» means that the i-th dancer leaves the stage. The substring «++i» means that the i-th dancer leaves while the similar $(i+1)$-th dancer appears, and «--i» means that the i-th dancer leaves while the similar $(i-1)$-th dancer appears. The output will be processed from the left to the right.

If there are many solutions, you may output any of them. If there is no solution, print 0 to the only line of output.

Examples

input.txt	output.txt
2 1	+1++1-2

Problem K. Wiki Lists

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	64 megabytes

In this task you need to write a small part of wiki-to-HTML translator. You have to deal only with enumerated and regular lists.

Wiki dialect in this task defines lists as follows — the character "#" defines an enumerated list and "*" defines a regular list. These symbols are called list markers.

Wiki-text should be processed line by line. A group of two or more consecutive lines should be treated as elements of the same list (enumerated, or regular) if these lines start with the same list marker symbol.

A group of two or more consecutive list elements should be considered as a nested list if these elements start with the same list marker symbol. This rule should be applied recursively.

HTML equivalent of a wiki-text is formed using the well-known HTML syntax. An enumerated list starts with "
", then all list elements are written, and then the list ends with "
". HTML equivalent of a regular list is similar to the one for enumerated list, but it starts with "
" and ends with "
". An element of any list is represented by "
", then the contents of the element are written followed by "
".

The rules given above unambiguously define the way to make HTML from wiki-text. Please refer to the sample tests for clarifications.

Input

Input contains a wiki-text with no less than 1 and no more than 1000 lines. The total length of all input lines is not greater than 1000 characters. The depth of nested lists is not limited. Input contains only lowercase and uppercase Latin letters, digits, symbols "*" and "#", and line breaks. Each line contains at least one character different from "*" and "#", so there will be no empty lines in the output.

Output

Print HTML equivalent of the given wiki-text to the output. All tags should be placed on separate lines with no spaces.

Examples

input.txt	output.txt
FirstLine *ItemX *ItemY #Item1 #Item2 *ItemZ	FirstLine ItemX ItemY Item1 Item2 *ItemZ

input.txt	output.txt
*ab *x#x *#1 *#2 #3	 ab x#x 1 2 #3

input.txt	output.txt
***1 **2	 *1 2