Problem A. Bridging the Gap 2

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

A group of n walkers arrives at a riverbank at night. They want to cross the river using a boat, which is initially on their side. The boat can hold at most R walkers at a time and requires at least L ($1 \le L < R$) walkers to operate.

Rowing the boat is a tiring task. Each time the boat is used to transport a group of walkers to the other side of the river, all walkers on the boat must have stamina greater than 0, and each walker's stamina decreases by 1 after the trip. Initially, the i-th walker ($1 \le i \le n$) has a stamina value of h_i.

You need to determine if it is possible to transport all the walkers to the other side of the river using the boat.

Input

The first line of input contains three integers n, L, R ($1 \le L < R \le n \le 5 \times 10^5$), denoting the number of walkers, the minimum and the maximum number of walkers to use the boat at a time, respectively.

The second line of input contains n integers $h_1, h_2, \ldots, h_n$ ($1 \le h_i \le 5 \times 10^5$), where h_i ($1 \le i \le n$) denotes the stamina value of the i-th walker.

Output

If it is possible to transport all the walkers to the other side of the river using the boat, output "Yes" in the first line. Otherwise, output "No" in the first line. You can output each letter in any case (lowercase or uppercase). For example, the strings "yEs", "yes", "Yes", and "YES" will all be considered as positive replies.

Examples

standard input	standard output
4 1 2 1 2 5 10	Yes
5 1 2 1 1 1 1 5	No
5 1 3 1 1 1 1 5	Yes
5 2 3 1 1 1 1 5	No
9 1 9 1 1 1 1 1 1 1 1 1	Yes

Problem B. Crash Test

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	1024 megabytes

After five years, the most high-profile event in motor racing, Formula 1, returns to China. The Chinese Grand Prix was recently held at the Shanghai International Circuit. Formula 1 cars can reach speeds of up to 350 km/h. To ensure the safety of the drivers, the cars must pass rigorous crash tests.

We consider the following simplified version of a crash test. Initially, a car is positioned with its front facing a wall, at a distance of D meters from the wall. This crash test provides n types of boosters, where the i-th type of booster has a thrust performance of h_i, and there are ample quantities of each type of booster. Suppose the current distance between the car's front and the wall is d, and we use a booster with a thrust performance of h. When $d \geq h$, the car will move forward h meters and then stop. Otherwise, the car will move forward d meters, crash into the wall, and rebound $h - d$ meters, after which it stops, still facing the wall.

Now, you want to know, through any number of operations (including no operation), what the minimum distance between the car's front and the wall can be?"

Input

The first line of input contains two positive integers n and D ($1 \leq n \leq 100, 1 \leq D \leq 10^{18}$), denoting the number of boosters and the distance between the Formula 1 car and the wall, respectively.

The second line of inputcontains n positive integers $h_1, h_2, \ldots, h_n$ ($1 \leq h_i \leq 10^{18}$), denoting the thrust performance of each booster.

Output

Output an integer in a line, denoting the minimum possible distance between the car's front and the wall.

Examples

standard input	standard output
1 10 3	1
2 10 3 4	0
2 1 3 7	0
10 4306315981482911 4306 3519 8148 2911 9970 7222 5462 1844 7072 1702	0
10 123456789987654321 10 10 10 10 10 10 10 10 10 10	1

Note

We will use $x \overset{i=k}{\to} y$ to represent one operation, where x and y are the distances between the car's front and the wall before and after the operation, and $i = k$ indicates that the k-th booster is chosen for this operation.

For the first sample test, an optimal strategy is $10 \overset{i=1}{\to} 7 \overset{i=1}{\to} 4 \overset{i=1}{\to} 1$.

For the second sample test, an optimal strategy is $10 \overset{i=1}{\to} 7 \overset{i=2}{\to} 3 \overset{i=1}{\to} 0$.

For the third sample test, an optimal strategy is $1 \overset{i=2}{\to} 6 \overset{i=1}{\to} 3 \overset{i=1}{\to} 0$.

Problem C. Delete 4 edges

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	256 megabytes

Given two connected graphs G_1, G_2. You should count the number of ways to delete 4 edges from $G_1 \square G_2$ such that $G_1 \square G_2$ becomes disconnected, where $G_1 \square G_2$ is the Cartesian product of G_1 and G_2 (See the Notes section for formal definitions).

Since the answer might be large, you should output the answer modulo 998 244 353.

Input

The input consists of descriptions of the two graphs G_1 and G_2 sequentially.

For each graph, the first line contains two integers n, m ($5 \le n \le 2 \cdot 10^5, n - 1 \le m \le 2 \cdot 10^5$), denoting the number of vertices and edges in the graph, followed by m lines, with each line containing two integers u, v ($1 \le u, v \le n$), denoting an edge in the graph.

It is guaranteed that both graphs are connected, and have no self-loop or multiple edges.

Output

Output an integer in a line, denoting the number of ways to delete 4 edges to disconnect the graph $G_1 \square G_2$, taken modulo 998 244 353.

Examples

standard input	standard output
5 5 1 2 2 3 3 4 4 5 5 1 6 6 1 2 2 3 3 4 4 5 5 6 6 1	30
6 5 1 2 1 3 1 4 2 5 2 6 5 5 1 2 2 3 3 4 4 1 3 5	6154

Note

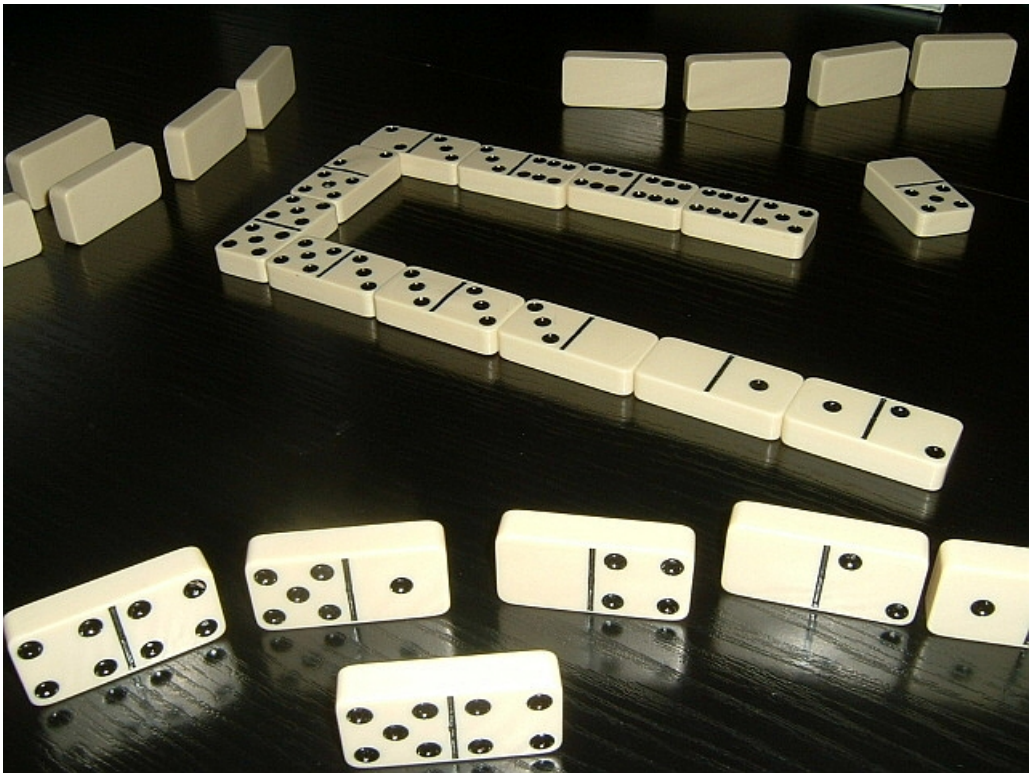
In graph theory, the Cartesian product $G\square H$ of graphs G and H is a graph such that:

- the vertex set of $G\square H$ is the Cartesian product $V(G) \times V(H)$; and
- two vertices (u, v) and (u', v') are adjacent in $G\square H$ if and only if either
  - $u = u'$ and v is adjacent to v' in H, **or**
  - $v = v'$ and u is adjacent to u' in G.

Problem D. Dominoes!

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Dominoes are rectangular tiles made of wood, bone, or plastic. They are arranged in a line at a certain distance from each other, and when the first tile is lightly tipped over, the rest of the tiles fall in succession, creating a chain reaction.



However, now you want to use the dominoes to play a different game. Given n dominoes, each domino has two positive integers between 1 and 10^9 written on it. You need to arrange these dominoes in a horizontal line so that each end of a domino matches the end of the adjacent domino (except for the two ends of the line). When arranging the dominoes, you can choose either number to be on the left or right. The task is to determine if there is an arrangement where the numbers at the touching ends of adjacent dominoes are always different. If such an arrangement exists, provide a possible solution.

Input

The first line of input contains an integer n ($1 \le n \le 2 \times 10^5$), denoting the number of dominoes.

Then n lines follow. The i-th ($1 \le i \le n$) line contains two integers x_i, y_i ($1 \le x_i, y_i \le 10^9$), denoting the numbers written on the i-th domino.

Output

If there exists a valid way to arrange dominoes, output "Yes" in the first line. Otherwise, output "No" in the first line. You can output each letter in any case (lowercase or uppercase). For example, the strings "yEs", "yes", "Yes", and "YES" will all be considered as positive replies.

If your answer is "Yes", you need to output two numbers x_i, y_i in the next n lines, where the i-th ($1 \le i \le n$) line represents the numbers on the left and right sides of the i-th domino in your solution, arranged from left to right.

Examples

standard input	standard output
3 1 2 2 3 3 1	Yes 1 2 3 2 3 1
3 1 1 1000000000 1000000000 1000000000 1000000000	Yes 1000000000 1000000000 1 1 1000000000 1000000000
2 2 2 2 2	No

Problem E. Malfunctioning Typewriter

Input file:	standard input
Output file:	standard output
Time limit:	3 seconds
Memory limit:	1024 megabytes

Recently, Bobo, having some free time, wants to write a poem. His method of writing poems is straightforward: he carefully selects n verses, where each verse is a binary string of length m and all n verses are distinct. He then considers **any** concatenation of these n verses in any order to form a binary string of length nm as a good poem.

Unfortunately, Bobo only has a broken typewriter. When Bobo tries to type x (where x is either 0 or 1) on paper, the typewriter has a probability p ($0.5 \leq p < 1$) of typing x correctly and a probability $1 - p$ of typing $1 - x$.

Bobo wants to know, using the optimal strategy, what is the maximum probability that he can write a good poem?

Input

The first line of input contains three integers n, m, p ($1 \leq n \leq 1000, 1 \leq m \leq 1000, 0.5 \leq p < 1$), which represent the number of verses, the length of each verse, and the probability that the typewriter works correctly, respectively. The probability p is guaranteed to have at most six decimal places.

Then n lines follow, with the i-th line ($1 \leq i \leq n$) containing a binary string of length m, representing the content of the i-th verse. It is guaranteed that the n verses are distinct.

Output

Output a single number in a line, representing the maximum probability that Bobo can write a good poem using the optimal strategy. Your answer will be considered correct if and only if its absolute or relative error does not exceed 10^{-9}. Formally, if the correct answer is a and your output is b, then your answer will be considered correct if and only if $\frac{|a-b|}{\max(b,1)} \leq 10^{-9}$.

Examples

standard input	standard output
1 3 0.5 000	0.125000000000
2 3 0.9 010 101	0.590490000000
4 2 0.618 00 01 10 11	0.201586731534

Note

In the first sample test, the only good poem that meets Bobo's requirements is 000. In this case, every time Bobo presses 0 or 1, there is always a 0.5 probability of getting the desired 0. Under the optimal strategy, there is a probability of $(0.5)^3 = 0.125$ of producing a good poem.

In the second sample test, if the first character produced by the typewriter is 0, the only possible good poem is 010101. In this case, Bobo's optimal strategy is to press the corresponding keys for 010101 for the next positions, resulting in a success probability of $(0.9)^5 = 0.59049$. If the first character produced is 1,

the only possible good poem is 101010. In this case, Bobo's optimal strategy is to press the corresponding keys for 101010 for the next positions, also resulting in a success probability of $(0.9)^5 = 0.59049$. Thus, Bobo's optimal strategy is to first press any key, and then follow the corresponding good poem, with a success probability of $(0.9)^5 = 0.59049$.

Problem F. MMR Double Down Tokens

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	256 megabytes

Bobo has recently been playing Dota2. Due to a new event, "Crownfall" in Dota2, Bobo received a number of k Double Down Tokens. In ranked contests of Dota2, players receive $+1$ MMR for wins and -1 MMR for loses. MMR Double Down tokens are items that can be used **at the beginning of a ranked contest**, so that the MMR gained in this contest will be doubled ($+2$ for wins and -2 for loses).

However, Bobo suspects that Dota2 uses the ELO matching system to balance the winrates of players so that he will receive exactly n wins and m loses in the upcoming $n+m$ contests, and all possible combinations of n wins and m loses will **appear with equal probability**.

Now, Bobo wants to design a strategy to use some of the k MMR Double Down tokens so that after the $n+m$ contests, the expected additional MMR gained using MMR Double Down Tokens(that is, the MMR change across all $n+m$ contests, subtracted by $n-m$) is maximum. You need to find out this maximum expected value.

Input

The only line of input contains 3 integers n, m, k ($1 \le n, m \le 5 \cdot 10^6, 0 \le k \le n+m$).

Output

Output an integer in one line, denoting the answer. Under the input constraints of this problem, it can be shown that the answer can be written as $\frac{P}{Q}$, where P and Q are coprime integers and $Q \not\equiv 0 \pmod{998244353}$. You need to output $P \cdot Q^{-1} \pmod{998244353}$ as an answer, where Q^{-1} is the modular inverse of Q with respect to 998244353.

Examples

standard input	standard output
1 1 2	499122177
1 3 0	0
4 1 5	598946615
1 9 5	299473306

Note

For the first sample test, the optimal strategy of Bobo is as follows: if the first game is a lost, Bobo will use the MMR Double Down token at the beginning of the second contest, so he will guarantee an additional MMR gain of $+1$; if the first game is a win, Bobo will not use any MMR double down token. The expected additional MMR gained using this strategy is $\frac{1}{2} \times 1 + \frac{1}{2} \times 0 = \frac{1}{2}$.

Problem G. Pythagorathean Transformation

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	256 megabytes

For a positive integer s, you can perform the following operation arbitrary number of times:

1. Choose a positive integer triplet (a, b, c) such that $s \in \{a, b, c\}$, $a^2 + b^2 = c^2$ and $c \leq 10^{18}$.
2. Choose a positive integer $t \in \{a, b, c\} \setminus \{s\}$,then replace s with t.

Given S and T, you need to transform S into T within 2000 steps or determine that it is not possible.

Input

The only line of input contains two integers S, T ($1 \leq S, T \leq 10^9, S \neq T$).

Output

If it is not possible to transform S into T within 2000 steps, output a -1 in a line.

Otherwise, you need to output an integer m ($1 \leq m \leq 2000$) in the first line, denoting the number of steps.

Then you need to output m lines, with the i-th line containing a positive integer x ($1 \leq x \leq 10^{18}$), representing the result after the i-th operation.

Examples

standard input	standard output
3 5	1 5
4 12	2 5 12
1 100	impossible

Problem H. Rectangle Intersection

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	1024 megabytes

In an n by m grid, there are k rectangles, each described by a tuple (x_1, y_1, x_2, y_2) $(1 \le x_1 \le x_2 \le n, 1 \le y_1 \le y_2 \le m)$, indicating that the rectangle covers all cells from row x_1 to x_2 and from column y_1 to y_2. For a collection of rectangles, their intersection is defined as the set of all cells that are covered by all these rectangles simultaneously. If no cell is covered by all these rectangles simultaneously, we consider the intersection of these rectangles to be empty.

Now, you need to select some rectangles from the given k rectangles **such that their intersection is non-empty**, and you should minimize the number of cells contained in the intersection of these rectangles.

Input

The first line of input contains three positive integers n, m, and k $(1 \le n, m \le 2000, 1 \le k \le 10^6)$, representing the number of rows and columns in the grid, and the number of rectangles, respectively.

Then k lines follow, each containing a description of one rectangle. Each rectangle's description consists of 4 numbers x_1, y_1, x_2, y_2 $(1 \le x_1 \le x_2 \le n, 1 \le y_1 \le y_2 \le m)$, indicating that the rectangle covers all cells from row x_1 to x_2 and from column y_1 to y_2.

Output

Output an integer in a line, denoting the answer.

Examples

standard input	standard output
6 4 2 2 2 4 3 3 2 5 3	4
19 19 4 1 9 10 9 9 10 9 19 11 1 11 10 10 11 19 11	10
7 7 2 1 1 7 7 2 2 6 6	25
5 5 2 1 1 4 4 2 2 5 5	9

Problem I. Removing Elements

Input file:	`standard input`
Output file:	`standard output`
Time limit:	15 seconds
Memory limit:	1024 megabytes

Let $T \subseteq \{1, 2, \ldots, n\}$ be a set of integers. Suppose you have a set $S = \{1, 2, \ldots, n\}$. You may perform the following operations an arbitrary number of times (including zero times):

- Choose $k \in T$ such that $k \leq |S|$;
- Remove the k-th largest element in S.

Find the total number of sets S that can be generated through the process. The answer might be enormous, and you should output the desired value modulo 998 244 353.

Input

The first line of input contains an integer n ($1 \leq n \leq 200\,000$).

The second line of input contains a binary string $s = s_1 s_2 \ldots s_n$ of length n, denoting the set $T = \{1 \leq i \leq n \mid s_i = 1\}$.

Output

Output an integer in a line, denoting the answer.

Examples

standard input	standard output
3 101	6
5 00001	2
10 1011011000	695

Problem J. Rigged Games

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	1024 megabytes

In a two-player competitive match, for any positive integer a, we define the Best of $2a-1$ format as follows: the two players continue to compete until one of them wins a times, thereby winning the overall match. The Best of $2a-1$ format includes a maximum of $2a-1$ games and a minimum of a games.

For any two positive integers a and b, we define the Best of $2b-1$ of Best of $2a-1$ format as follows: the two players compete in a major match using the Best of $2b-1$ format, which consists of a maximum of $2b-1$ major games and a minimum of b major games. Each major game is a Best of $2a-1$ format, consisting of a maximum of $2a-1$ minor games and a minimum of a minor games. For example, Team A and Team B play a Best of 3 of Best of 5 match. Here are some possible outcomes (we use 1 to represent a win for Team A and 0 to represent a win for Team B):

- 000111000 (Team A loses the first and third major games 0:3, wins the second major game 3:0, so the overall score is a 1:2 loss for Team A);
- 0110000110 (Team A loses the first and second major games 2:3, so the overall score is a 0:2 loss for Team A).

Team A and Team B are currently engaged in a DotA2 match using the Best of $2b-1$ of Best of $2a-1$ format. To the spectators, this is an exciting event! However, unknown to the spectators, the entire match outcome is predetermined: there is a "script"of length n represented by a binary string T, where the results of the minor games will continuously repeat this pattern of 01. Having this information, you want to know the match results when the "script"01 string is repeated from each position (see the sample explanation for details).

Input

The first line of input contains three integers n, a, b ($1 \le n, a, b \le 10^5$), whose meanings are already given in the statement.

The second line of input contains a binary "script" string T with length n.

Output

Output a binary string ans of length n, where $ans[i]$ ($1 \le i \le n$) represents the final match result when repeating the script string T starting from the i-th position of T.

Examples

standard input	standard output
3 1 1 001	001
7 2 2 1010101	1110111
10 5 3 1001011001	0011010011
10 2 4 1010000011	0000011110
2 100000 100000 01	01

Note

For the second sample test, the situation starting from each position is as follows:

Starting from the first position, the match results in $1010101\cdots$, and the final match sequence is 10101011. Team A wins 2:1 in major matches, with scores of 2:1, 1:2, 2:0, respectively.

Starting from the second position, the match results in $0101011\cdots$, and the final match sequence is 010101101. Team A wins 2:1 in major matches, with scores of 1:2, 2:1, 2:1, respectively.

Starting from the third position, the match results in $1010110\cdots$, and the final match sequence is 101011. Team A wins 2:0 in major matches, with scores of 2:1, 2:1, respectively.

Starting from the fourth position, the match results in $0101101\cdots$, and the final match sequence is 01011010. Team A loses 1:2 in major matches, with scores of 1:2, 2:0, 1:2, respectively.

Starting from the fifth position, the match results in $1011010\cdots$, and the final match sequence is 101101. Team A wins 2:0 in major matches, with scores of 2:1, 2:1, respectively.

Starting from the sixth position, the match results in $0110101\cdots$, and the final match sequence is 011010101. Team A wins 2:1 in major matches, with scores of 2:1, 1:2, 2:1, respectively.

Starting from the seventh position, the match results in $1101010\cdots$, and the final match sequence is 11010101. Team A wins 2:1 in major matches, with scores of 2:0, 1:2, 2:1, respectively.

Problem K. Slay the Spire: Game Design

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Bobo recently became obsessed with *Slay the Spire*, a deck-building roguelike game developed by Mega Crit. In Slay the Spire, players climb The Spire by ascending its floors and encountering various enemies, bosses, and events along the way. The map of Slay the Spire can be represented as a **directed acyclic graph**. Players can start at any *source vertex*, which are vertices with no incoming edges, and follow paths to reach any *sink vertex*, which are vertices with no outgoing edges and where bosses await. The map is designed in such a way where

- each vertex has at least one outgoing edge or at least one incoming edge (no isolated vertices);
- each vertex can reach at least one sink vertex, and can be reached from at least one source vertex through the edges.

After several games of Slay the Spire, Bobo becomes unsatisfied with simply playing, and now he wants to design the game by himself! The most important thing in game design is to control the difficulty of the game. So the first problem Bobo faces is to place the **minimum** number of elite enemies on some vertices (excluding source vertices and sink vertices) of the map so that no sneaky player can choose a path that avoids encountering at least k elite enemies. The formal definition of the problem is as follows:

> You are given a directed acyclic graph $G = (V, E)$ with no isolated vertices and a parameter k. Let $S \subseteq V$ be the set of vertices with no incoming edges, and $T \subseteq V$ be the set of vertices with no outgoing edges. You need to select a subset of vertices $U \subseteq V \setminus (S \cup T)$ such that any path P from some $s \in S$ to some $t \in T$ satisfy $|V(P) \cap U| \geq k$. Here $V(P)$ represents the set of vertices visited in P. If such a subset of vertices U exists, find any with the minimum size. Otherwise, report there is no such U.

Input

The first line of input contains three integers n, m, k ($2 \leq n \leq 1000, 1 \leq m \leq 5000, 1 \leq k \leq 5$), denoting the number of vertices, the number of edges and the parameter, respectively.

Then m lines follow, with each line containing two vertices u, v ($1 \leq u < v \leq n$), denoting a directed edge in the graph. It is guaranteed that this graph contains no repeated edges; each vertex has at least one outgoing edge or at least one incoming edges; each vertex can reach at least one sink vertex, and can be reached from at least one source vertex through the edges..

Output

If no valid subset of vertices exists, output -1 in a line. Otherwise, output a nonnegative integer ℓ in a line, denoting the size of the subset of vertices you choose. Then output ℓ distinct numbers $v_1, v_2, \ldots, v_\ell$ ($1 \leq v_i \leq n$) in a line, denoting the subset of vertices you choose. If there are multiple valid subset of vertices with the smallest size, outputting any of them will be considered correct.

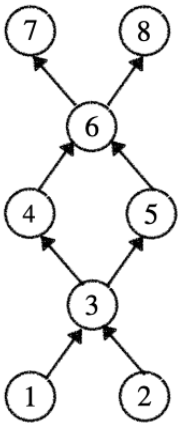
Examples

standard input	standard output
3 2 1 1 2 2 3	1 2
3 2 2 1 2 2 3	-1
8 8 2 1 3 2 3 3 4 3 5 4 6 5 6 6 7 6 8	2 3 6
6 14 1 1 2 1 3 2 3 1 4 2 4 3 4 1 5 2 5 3 5 4 5 2 6 3 6 4 6 5 6	4 2 3 4 5
15 14 2 1 2 1 3 2 4 2 5 3 6 3 7 4 8 4 9 5 10 5 11 6 12 6 13 7 14 7 15	6 2 3 4 5 6 7

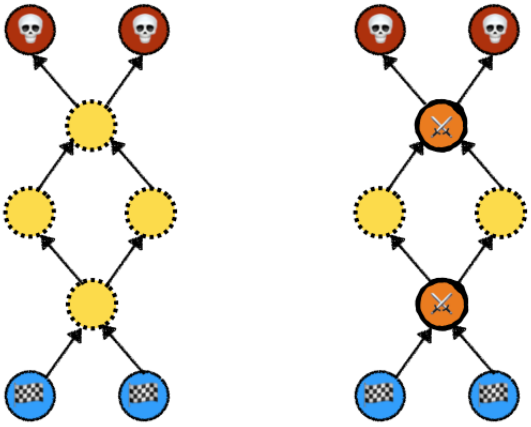
Note

Here, we provide graphical illustrations for the third sample test to help with understanding. The graph

given in the input is shown as follows.

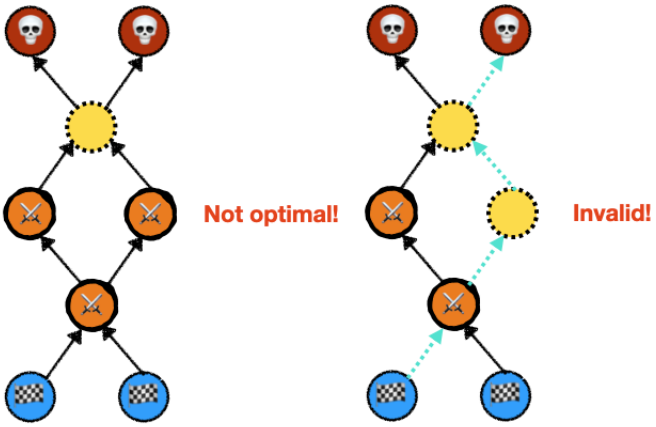


In this graph, the set of source vertices is $\{1, 2\}$, and the set of sink vertices is $\{7, 8\}$. We can place elite enemies at the remaining set of vertices, namely $\{3, 4, 5, 6\}$. The only optimal solution is to place elites on vertex 3 and vertex 6. In the following picture, we mark source vertices by flags, sink vertices by skeletons (representing bosses in-game), and elite enemies by swords.



The distinct types of vertices (left) and the only optimal solution (right)

For example, choosing vertices $\{3, 4, 5\}$ is also valid. However, this solution is not optimal as it has a size of 3. Choosing vertices $\{3, 4\}$ is not valid, as then there would exist a path from a source vertex to a sink vertex, encountering only one elite enemy. These two examples are shown in the following picture.



An solution that is not optimal (left) and a non-solution (right)

Problem L. Sudoku and Minesweeper

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Sudoku is a logic-based, combinatorial number-placement puzzle. In classic Sudoku, the objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 subgrids that compose the grid contains all of the digits from 1 to 9. The puzzle setter provides a partially completed grid, which for a well-posed puzzle has a single solution.

Minesweeper is a logic puzzle video game. The game features a grid of clickable tiles, with hidden mines scattered throughout the board. The objective is to clear the board without detonating any mines, with help from clues about the number of neighboring mines in each field.

You are given a solved classic Sudoku. You are asked to replace some numbers with mines so that the remaining numbers equal the number of neighboring mines. You can't replace all the numbers with mines.

Input

The input contains 9 lines. Each line contains 9 characters. Each character is a number between '1' and '9'. The input describes the solved classic Sudoku.

Output

Output 9 lines, denoting the answer. Each line contains 9 characters. Each character is either a number between '1' and '9' or '*'(which represents a mine). If there exist multiple solutions, output any.

Example

standard input	standard output
123647859	123******
648295317	*4*******
795831642	**5***642
436578921	**65***21
981462573	****6*5*3
257913486	*****34**
319754268	****54*6*
572386194	*****6**4
864129735	*********