Problem A. Chips

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	512 mebibytes

Programmer Vasiliy bought k cans of chips, with n chips in each can. Now he is watching a movie and eating chips. After each minute of the movie, our hero takes one of the cans and eats a chip from it.

At a certain moment, Vasiliy took one of the cans, but there were no chips left in it. What is the minimum and maximum number of minutes that could have passed since the start of the movie?

Input

The input consists of a single line containing two integers k and n separated by a space: the number of cans of chips and their size, respectively ($1 \le k, n \le 50$).

Output

Output two integers separated by a space: the minimum and maximum number of minutes that could have passed before Vasiliy took a can without chips.

Examples

standard input	standard output
3 4	5 13
49 15	16 736

Problem B. Application List

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	512 mebibytes

In a distant land, there lived a wise elder who was the keeper of all computer programs that ever existed. One day, the ruler of the land decreed to create a list all the programs, and the elder decided to do it by creating an Application List that would display the presence or absence of various applications.

He turned to you for help in this important task. You are given a list of programs with names consisting of lowercase English letters.

Your task is to fill a table of five rows and six columns. The first 26 cells of the table correspond to letters "a", "b", . . . , "z" in alphabetic order. The last four cells are empty. Each cell contains the corresponding letter if there is at least one program in the list whose name starts with that letter. If there are no such programs, a dot should be placed instead.

Input

The first line contains an integer n: the number of programs ($0 \le n \le 100$). Next, there are n lines listing the program names. Each name contains from 1 to 20 lowercase English letters.

Output

Output the Application List in the format described above: five lines with letters and dots, without spaces between them. The first four lines must contain six characters each. The last line must contain two characters.

Example

standard input	standard output
10 apple banana cherry date eggplant fig grape kiwi lemon mango	abcdef g...kl m..... ....

Problem C. Ordinal Number

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

Ordinal numbers are an extension of the set of nonnegative integers. For each nonnegative integer x, we will establish the corresponding ordinal number $f(x)$. The first few ordinal numbers can be defined as follows.

- Zero corresponds to an empty set:
  $f(0) = \{\}$.

- One corresponds to the set containing the set $f(0)$ as an element:
  $f(1) = \{f(0)\} = \{\{\}\}$.

- Two corresponds to the set containing the sets $f(0)$ and $f(1)$ as elements:
  $f(2) = \{f(0), f(1)\} = \{\{\}, \{\{\}\}\}$.

- And so on: each positive integer k corresponds to the set containing all the previous ordinal numbers as elements. The formula is:
  $f(k) = \{f(0), f(1), \ldots, f(k-1)\}$.

Next, we can similarly define ordinal numbers that don't correspond to integers. Alas, we won't need them in this problem.

You are given a string describing an ordinal number corresponding to a nonnegative integer n. Find n.

Input

The first line contains the description of an ordinal number corresponding to a certain nonnegative integer n ($0 \le n \le 15$). It consists of the characters "{", ",", and "}".

In the description of each set, each element appears exactly once. However, as a set does not change if we change the order of elements, this order can be arbitrary.

Output

Print the integer n corresponding to the given ordinal number.

Examples

standard input	standard output
`{}`	0
`{{}}`	1
`{{},{{}}}`	2
`{{{}},{{{}},{}},{}}`	3

Problem D. RACI

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

Nikolai assigned students the task of creating a RACI matrix for a project during management lectures. This is a responsibility assignment matrix that lists all stakeholders of the project and their levels of involvement in different tasks. The levels are denoted by the letters "R", "A", "C", and "I". If there is no involvement, "-" is used. The levels of involvement have the following meaning:

- R (*Responsible*): performs the task (if they are absent, then *Accountable* performs the whole task)

- A (*Accountable*): accepts the task from *Responsible*; for each task, there must be exactly one instance of this level of involvement, unlike the other levels, of which there can be any number

- C (*Consulted*): provides consultation during the execution of the task

- I (*Informed*): receives information about the progress of the task

Help the students verify the correctness of the matrix.

Input

The first line contains two integers n and m: the number of rows and columns of the RACI matrix ($1 \leq n, m \leq 100$).

Next, n rows are listed, each containing m elements separated by spaces.

Each row represents a task, and each column corresponds to a stakeholder.

Each element of the matrix can be either an uppercase letter "R", "A", "C", or "I", or a minus sign, "-", indicating that the given stakeholder has no level of involvement in this task.

Output

Print "Yes" if the matrix is correct, or "No" otherwise. Letter case does not matter.

Examples

standard input	standard output
3 5 C C A - I A R - C I A R I C -	Yes
3 3 A - C R C I - A I	No

Problem E. Triangle on the Axis

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	512 mebibytes

You are given a set of points on a plane with integer coordinates. Find a triangle with the largest area whose vertices belong to this set of points, with one of its sides lying on the Ox axis.

Input

The first line contains an integer n: the number of points ($1 \leq n \leq 1000$). Each of the following n lines contains two integers x and y: the coordinates of the points. All coordinates do not exceed 1000 by absolute value.

Output

Output a single real number: the maximum area of the triangle that satisfies the problem's conditions. If there is no such triangle or it is degenerate, output 0.
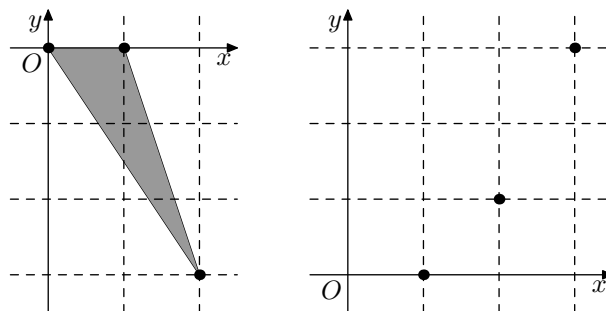
Your answer will be considered correct if it differs from the exact value by no more than 10^{-9}.

Examples

standard input	standard output
3 0 0 1 0 2 -3	1.5
3 1 0 2 1 3 3	0

Note

Pictures for the examples:

Problem F. Faulty Traffic Light

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	512 mebibytes

Near Vlad's home, there is a pedestrian traffic light that shows the remaining time to cross the road. The number displayed by the traffic light decreases by 1 every second. However, one day, Vlad approached the crossing and could not believe his eyes: the traffic light switched from 35 to 36!

Vlad was very surprised, but his astonishment faded in the next second: instead of 37 or any other number, the traffic light displayed something that was not a number at all. It turned out that one of the indicators of the traffic light was broken and never lit up. Hence, the number 39 looked like 35, and 38 looked like 36 (see Figures 1–3). After that, Vlad wondered: just how long can this illusion of the traffic light "ticking up by 1" every second last?

Figures 1–3. The behavior of the traffic light from the moment it switched to 39 until the moment it switched to 37: expected is on the top, actual is on the bottom. Lit indicators are colored green (looks gray when printed in black and white), functioning but not currently lit indicators are colored white, and the broken indicator is hatched.

39 looks like 35   38 looks like 36   37 looks like nonsense
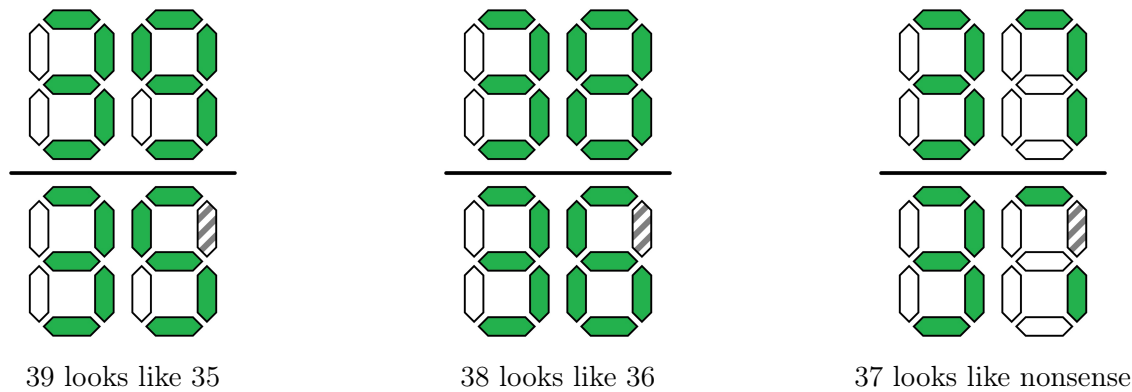
For example, in the above situation, the illusion has lasted for exactly 1 second: from the moment the traffic light "switched from 35 to 36" (actually from 39 to 38) until the moment the traffic light started showing nonsense. Strictly speaking, the traffic light has "ticked up by 1" for k seconds if it has consecutively displayed x, $x + 1$, ..., $x + k$.

You are faced with the following task. The traffic light shows two digits. Both digits are "drawn" in the standard way using seven "stick-like" indicators: four of them are vertical and three are horizontal (see Figure 4).
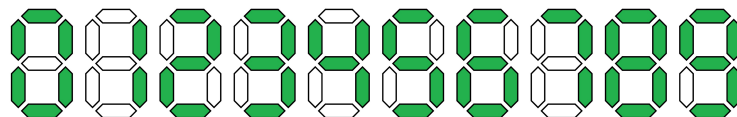

Figure 4. How the digits look on a functioning traffic light

Each indicator is either functioning (works as it should) or broken (never lights up). The color switching of the traffic light always works correctly and always destroys the illusion of "ticking up". You are given exactly 99 seconds to cross the road. Hence, a functioning traffic light consecutively shows 99, 98, 97, ..., 2 and 1. It **does not** show 0, and, instead, immediately switches to another color.

An important note: the traffic light **does not show leading zeros**, meaning that, on a functioning traffic light, the number 6 is displayed as a single six in the second digit, not as "06". Figures 5–6 show how the numbers 6 and 37 are displayed on a functioning traffic light.
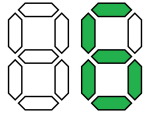
Figure 5. The number 6 on a functioning
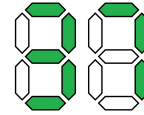traffic light

Figure 6. The number 37 on a functioning
traffic light

If such an illusion cannot be created at all (for example, when the traffic light always shows nonsense), then the answer is considered to be 0.

Summing up, the task can be formalized as follows. Find the largest non-negative k such that there exists at least one y such that $k + 1 \leqslant y \leqslant 99$ and the numbers y, $y - 1$, ..., $y - k$ look as the numbers x, $x + 1$, ..., $x + k$ due to the broken indicators. If such a number k does not exist, then the answer should be 0.

Input

The first line contains a string of seven characters, with each being either "0" or "1". They describe the states of the indicators in the first digit of the displayed number according to the order in Figure 7: the first character corresponds to the top horizontal indicator, the second to the upper left vertical indicator, the third to the upper right vertical indicator, the fourth to the middle horizontal indicator, the fifth to the lower left vertical indicator, the sixth to the lower right vertical indicator, and the seventh to the bottom horizontal indicator. A "1" means that the corresponding indicator is functioning, and a "0" means that it is broken.

The second line describes the state of the indicators in the second digit of the displayed number in a similar manner.
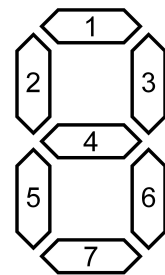

Figure 7. The order
of indicators

Output

Print a single integer: the answer to the problem.

Examples

standard input	standard output
1111111 1101111	1
1011011 1101111	1
1101111 0010010	0
0000000 0000000	0

Note

The first example is the situation discussed in the statement. The statement shows how the illusion can last for one second. It can be shown that it cannot last longer.

In the second example, additionally, the second and fifth (left vertical) indicators in the first digit are broken. The answer of 1 is still achieved in the same way (39 looks like 35, and 38 looks like 36).

In the third example, the answer is 0. Note that 90 on such a traffic light looks like 51, and 89 looks like 61, which is greater than 51. However, these two consecutive displays of the traffic light do **not** form an illusion of length 1, as the numbers 51 and 61 are not consecutive.

In the fourth example, all indicators are broken, and therefore, the traffic light never displays a correctly written number. According to the statement, in this case, the output should be 0.

Problem G. Programmers and Stones

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	512 mebibytes

Programmers Alice and Dmitry invented a new game. In this game, there are n piles of stones on the table. The players take turns starting from Alice. On their turn, a player picks an arbitrary non-empty set of non-empty piles, and then remove one stone from each of them. The player who can't make a move loses. Who will win the game if both play optimally?

Input

The first line contains an integer n ($1 \leq n \leq 100\,000$).

The second line contains n numbers $a_1, a_2, \ldots, a_n$: the initial sizes of the piles of stones ($1 \leq a_i \leq 10^9$).

Output

Print "Alice" or "Dmitry", depending on who wins the game. In the names, letter case **does** matter.

Examples

standard input	standard output
5 1 2 3 4 5	Alice
2 2 2	Dmitry

Problem H. Minimization by Swaps

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	512 mebibytes

Masha is studying large numbers. She has placed n cards in a row. Each card has a digit from 1 to 9 written on it. Together, they form an n-digit integer s.

In one operation, Masha can take two **adjacent** cards and swap them (she cannot rotate the cards, turning one digit into another). Masha can perform no more than k operations. What is the minimum n-digit number that can be obtained as a result?

Input

The first line contains an integer t: the number of test cases ($1 \le t \le 100\,000$). The following lines contain the test cases.

Each test case is given on a line containing two integers s and k separated by a space. The integer s is positive and consists of digits from 1 to 9. Additionally, $0 \le k \le 10^{18}$.

The total number of digits in all numbers s does not exceed $100\,000$.

Output

For each test case, print a line with the answer: the minimum n-digit number that can be obtained from s by swapping two adjacent digits no more than k times.

Example

standard input	standard output
4	321
321 0	9
9 1	11122247
21241127 10	629
692 1	

Problem I. Modular Taxi

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

Longlandia is a very long country. All of its n cities are located along a line segment. If we enumerate them from the beginning to the end of the segment, the i-th city has a_i inhabitants.

You need to get from city s to city f. For this purpose, an infinite number of taxis called Kaban-2, Kaban-3, Kaban-4, Kaban-5, ... operate in Longlandia. A taxi named Kaban-m can take you from city i to city j if the numbers of inhabitants in all cities from i to j inclusive are congruent modulo m. Formally, for any integer k such that $\min\{i, j\} \leq k \leq \max\{i, j\}$, the relation $a_k \equiv a_i \pmod{m}$ must hold.

Find the smallest number Q of taxi calls required to get from city s to city f, and output Q lines describing the route. If it is impossible to reach the destination by taxi, output "`Impossible`".

Input

The first line contains an integer n: the number of cities in Longlandia ($2 \leq n \leq 2 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$: the population of each city ($1 \leq a_i \leq 10^9$).

The third line contains two integers s and f: the starting and finishing city numbers ($1 \leq s, f \leq n; s \neq f$).

Output

Let Q be the smallest number of taxi calls required to get from city s to city f. Output Q lines of the form "`Kaban-`m_i s_i f_i", indicating that the i-th trip will be made by taxi Kaban-m_i and will take you from city s_i to city f_i ($1 \leq s_i, f_i \leq n; 2 \leq m_i \leq 10^9$). The following equalities must hold: $s_1 = s$; $f_Q = f$; $s_{i+1} = f_i$. And, of course, taxi Kaban-m_i must be able to take you from city s_i to city f_i.

If it is impossible to reach from s to f with any number of taxi calls, output the word "`Impossible`".

Letter case does not matter, so you can output, for example, "`kaBAN`" and "`IMPossiBle`".

Examples

standard input	standard output
6 1 2 3 4 5 6 5 3	Impossible
8 1 16 20 20 20 23 7 8 1 7	Kaban-5 1 2 Kaban-4 2 5 Kaban-3 5 6 Kaban-8 6 7
11 55 55 55 55 55 55 55 55 55 55 55 7 2	kaBAn-239239239 7 2

Problem J. Electrician

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

There are n houses with internet access in village J. In one of the houses, there lives a villain who is doing their villainous deeds: continuously writing us angry comments on social media from a fake account. We do not know which house they live in, but we want to find that out.

At the electrical substation, there are n switches, one for each house. When a switch is on, the light is on in the house, and when it is off, there is no light, and, by extension, no internet access. Initially, all switches are on. At the start of every hour, we can change the state of the switches in any way we wish: maybe turn some on and maybe turn some off. After that, for the next hour, we monitor social media: if the villain's house has light, comments will definitely appear, and if there is no light, there will be no comments. After our investigation, we must return everything to its original state: turn the light back on in all houses.

Unfortunately, if the light in the villain's house goes out more than k times, they will become suspicious, and we would like to avoid that. It is important to note that the light can safely be absent for a long time, it is the number of times we turn it off that counts. What is the minimum number of hours required for us to confidently determine which house the villain lives in and to restore the light in all houses?

Input

The first line contains an integer t: the number of test cases ($1 \le t \le 2 \cdot 10^5$). Then t test cases follow.

Each test case is given on a separate line containing two integers n and k: the number of houses and the maximum possible number of times we can switch the light off in the villain's house ($1 \le n, k \le 10^{18}$).

Output

For each test case, output a line with a single integer: the minimum number of hours required to determine the villain's house and restore the light for everyone.

Example

standard input	standard output
5	2
3 1	2
4 1	1
2 2	5
17 3	7
73 6	

Note

In the first test case, $n = 3$ and $k = 1$. We can start by turning off the light in houses numbered 1 and 2 during the first hour, then after an hour, we can restore the light in house number 1, and after another hour, restore the light in house number 2. By the beginning of the third hour, we will know for sure where the villain lives, and we will restore the light in all houses.

Problem K. Wooden Matrix

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

Consider a square matrix of size $n \times n$ consisting of non-negative integers. The matrix is symmetric with respect to the main diagonal, and the main diagonal itself contains only zeroes. Such a matrix is called *wooden* if there is an undirected tree T on n vertices with edges of positive lengths such that each cell (i, j) of the matrix contains the distance between vertices i and j in this tree.

You are given a matrix. Check if it is wooden.

Input

The first line contains an integer n: the size of the matrix ($1 \leq n \leq 1000$). Each of the following n lines contains n integers $d_{i,j}$: the elements of the matrix ($0 \leq d_{i,j} \leq 10^9$). The matrix is symmetric with respect to the main diagonal. There are zeros on the main diagonal and strictly positive integers outside it.

Output

Print "Yes" or "No" depending on whether the matrix is wooden. Letter case does not matter.

Examples

standard input	standard output
3 0 1 3 1 0 2 3 2 0	Yes
3 0 1 3 1 0 1 3 1 0	No

Problem L. Count the Operations

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	512 mebibytes

In a class, Ania wrote a program which does some work for all integers from 0 to $n - 1$:

```
for (int i = 0; i < n; i++) {
    work (i);
}
```

The teacher Petia looked at the program and said that it would work for too long, as the maximum value of n is 10^9.

In response, Ania decided to do the work only for a portion of the numbers. Her new program looks as follows:

```
for (int i = 0; i < n; i++) {
    if (i == x_1) i = y_1;
    if (i == x_2) i = y_2;
    ...
    if (i == x_k) i = y_k;
    work (i);
}
```

How many operations does this program perform for the given n, and does it terminate at all? The operations we count are assignments (i = 0; i++; i = y_1; ...), comparisons (i < n; i == x_1; ...), and the work itself (work (i)).

Input

The input contains the program written by Ania. It is formatted exactly as shown in the examples: in particular, the opening curly bracket is on the line with `for`, there is a space in front of every opening round bracket, and the indentation is four spaces. The input does not contain spaces at line ends.

The number of lines with conditionals: $0 \le k \le 10^5$. The constraints on the values: $0 \le x_i, y_i < n \le 10^9$, other than that, the values can be arbitrary. The line with `work` follows once after all the lines with `if`.

Output

Print the number of operations that the given program performs. If the program never terminates, print -1.

Examples

standard input	standard output	operations

`for (int i = 0; i < 3; i++) {`
`    work (i);`
`}`
	11	i = 0; 0 < 3; work (0);
		i++; 1 < 3; work (1);
		i++; 2 < 3; work (2);
		i++; 3 < 3.

`for (int i = 0; i < 5; i++) {`
`    if (i == 3) i = 1;`
`    if (i == 1) i = 3;`
`    if (i == 3) i = 4;`
`    work (i);`
`}`
	16	i = 0; 0 < 5; 0 == 3;
		0 == 1; 0 == 3; work (0);
		i++; 1 < 5; 1 == 3;
		1 == 1; i = 3; 3 == 3;
		i = 4; work (4); i++;
		5 < 5.

Problem M. Balls of Three Colors

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	512 mebibytes

There are r red, g green, and b blue balls. How many ways are there to arrange all these balls in a row such that any two adjacent balls have different colors? Since this number can be very large, output its remainder when divided by the prime number $998\,244\,353$.

Input

You are given three integers separated by spaces: r, g, and b. Each of the integers is from 1 to 10^5 inclusive.

Output

Output a single integer: the required number of ways modulo $998\,244\,353$.

Examples

standard input	standard output
1 1 1	6
4 1 1	0
1 1 2	6

Problem N. Definitely Not Chess

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

Your task is to checkmate the black king using the white king, a camel, and a wazir.

The camel (represented by an inverted knight symbol) is a chess piece that moves in an elongated knight's move: three squares in one direction (one more than a knight) and one square in the other. The wazir (represented by an inverted rook symbol) always moves to an adjacent square along a rank or file, in other words, makes rook moves of length 1. The king moves as in regular chess: to any adjacent square along a rank, file, or diagonal.

Below are the possible moves of all pieces from the e4 square. Squares reachable in one move are marked by a cross.
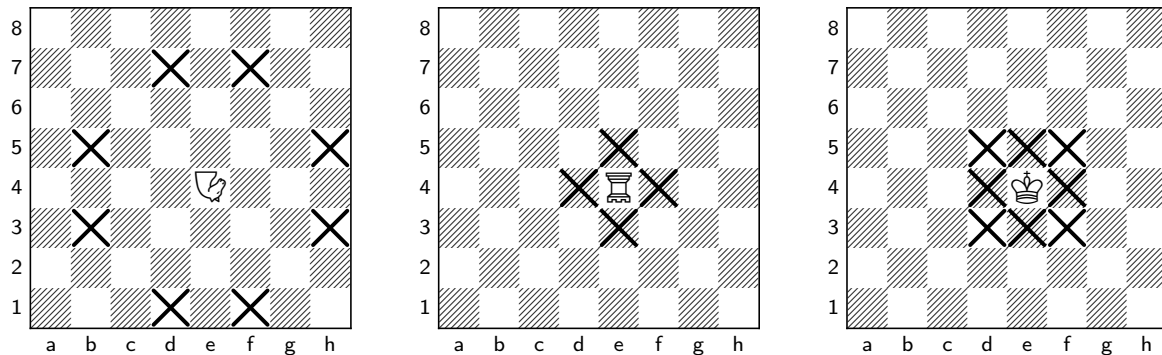
The rule of declaring a draw after 50 moves does not apply.

Input

The first line contains an integer t: the number of test cases ($1 \le t \le 2 \cdot 10^6$). Then t test cases follow.

Each test case is given on a separate line in the following format: squares of the white king, black king, white camel, and white wazir in algebraic notation, followed by a letter "w" or "b" indicating whether it is White's or Black's turn, respectively. Algebraic notation for chessboard squares consists of two characters: a file number encoded by a letter from "a" to "h", and a rank number encoded by a digit from "1" to "8". Ranks are numbered from bottom to top, and files from left to right, as shown in the pictures.

In each test case:

- All pieces are on distinct squares.

- The player that is not to move is not in check.

- The game is not over, meaning there cannot be a stalemate or a checkmate on the board.

Output

For each test case, print the evaluation of the position on a separate line. The evaluation of the position should be either the number 0 if the position is drawn, or the character "#" immediately followed by the minimum number of moves required to guarantee a checkmate.

Example

standard input	standard output
3	#1
f3 f1 f4 g2 w	0
f3 f1 e1 g2 b	#66
a1 c3 e5 h8 b	

Note

The positions in the example look as follows:
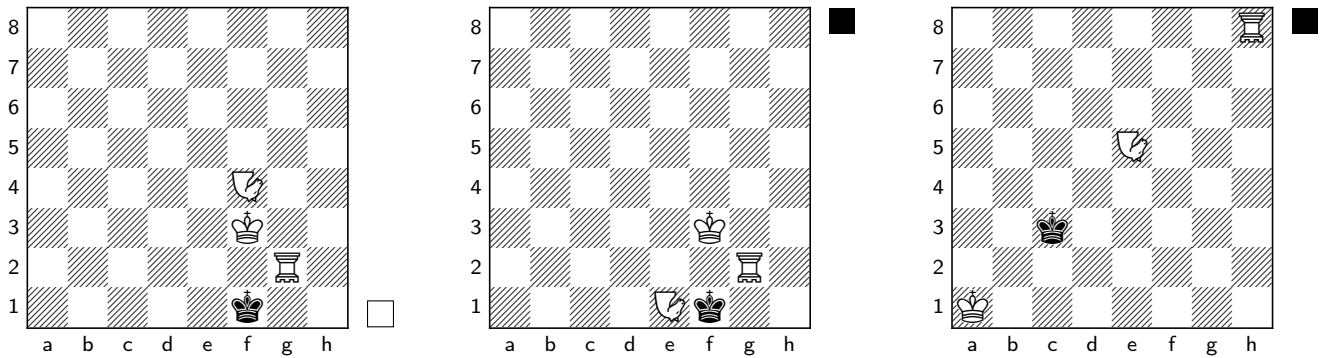
This problem assumes the following standard chess rules and concepts:

- Players alternate moves: one for black and the other for white pieces.

- During their turn, a player must choose one of their pieces and make a move with it.

- It is forbidden to move pieces to squares occupied by pieces of the same color.

- Moving to a square occupied by a piece of the opposite color is called *capturing*. The captured piece is removed from the game.

- *Check* is a state when one of the pieces can move to the square occupied by the king of the opposing color.

- A player is forbidden from making a move that puts their king in check at the end of the turn.

- A state where the player with the turn has no valid moves and their king is **not** in check is called *stalemate* and results in a draw.

- A state where the player with the turn is in check and has no valid moves is called *checkmate* and is declared a victory for their opponent. In this problem, only White can achieve a victory.

- When counting the number of moves until checkmate, only the moves of the checkmating side are considered.

- Positions from which the game could proceed indefinitely with optimal play are considered drawn in this problem.