

# Forgery

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

Camille is attending an auction to acquire some new artworks for their gallery. The auction only offers black-and-white artworks on a grid. However, they suspect that some of the artworks may be forged!

Luckily, any reputable artist would sign their name on their work, so Camille plans to inspect the art closely to determine whether the signature is present. Given the signature and the full artwork, help Camille decide whether the artwork is genuine or a forgery.

## Input

The first line of input consists of four space-separated integers  $r_1$ ,  $c_1$ ,  $r_2$  and  $c_2$  ( $1 \leq r_1 \leq r_2 \leq 100$ ,  $1 \leq c_1 \leq c_2 \leq 100$ ), representing the number of rows and columns in the signature and the artwork respectively.

$r_1$  lines follow, the  $i$ th of which contains a string  $s_{i,1}s_{i,2}\dots s_{i,c_1}$ , where the  $j$ th character is either ‘.’ or ‘\*’, denoting a white or black pixel respectively in the signature.

$r_2$  lines follow, the  $i$ th of which contains a string  $s_{i,1}s_{i,2}\dots s_{i,c_2}$ , where the  $j$ th character is either ‘.’ or ‘\*’, denoting a white or black pixel respectively in the artwork.

## Output

Output one line, containing only the word “**Genuine**” (without punctuation) if the signature appears in the artwork, or “**Forgery**” (without punctuation) otherwise.

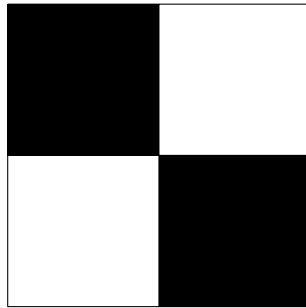
The signature must appear completely unchanged in the artwork, that is, it must be contiguous, not enlarged, and not rotated.

## Examples

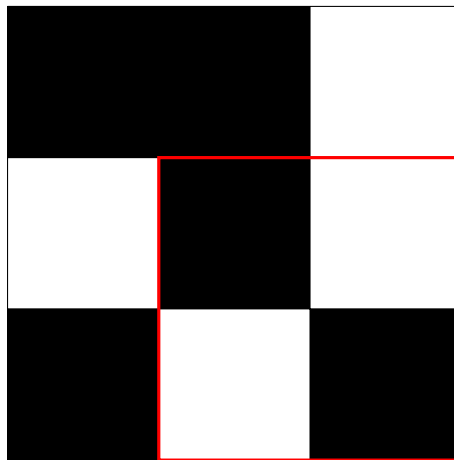
standard input	standard output
2 2 3 3 *. .* **. *. *.*	Genuine
1 2 2 3 ** *.* *.*	Forgery
2 2 2 2 *. .* .* *.	Forgery
2 2 2 2 ** *. ** *.	Genuine

## Note

In the first example, the artist's signature is



and the artwork is



featuring the signature in the bottom-right corner.

# Connect

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

Ayumi and Bunji are playing a game called Connect. They take turns to place red and blue tokens respectively on an  $n$  by  $n$  grid, with Ayumi always playing first.

The rules of the game are as follows.

- To place a token, the player need only select a column; the token must be placed in the bottom-most square of that column which does not already contain a token. Tokens cannot be added to a column where all  $n$  squares already contain tokens.
- There is a target number  $k$ . The first player to make a horizontal, vertical or diagonal line of  $k$  tokens in their colour wins the game.

Ayumi and Bunji have been playing this game for some time. They have meticulously written down the moves that were made, but they forgot to check whether the game has already been won, and by who. Help them determine the result of the game.

## Input

The first line of input consists of three space-separated integers  $n$  ( $1 \leq n \leq 300$ ),  $m$  ( $0 \leq m \leq n^2$ ) and  $k$  ( $1 \leq k \leq n$ ), representing the dimension of the grid, the number of moves played and the target number respectively.

The second line of input consists of  $m$  space-separated integers, the  $i$ th of which is the column selected on the  $i$ th move. Each of these integers is between 1 and  $n$  inclusive, and you are guaranteed that no column appears more than  $n$  times.

## Output

If either player has won the game, print a single line consisting of the winning player's name and the move number on which they won.

Otherwise, print the phrase "No winner" (without punctuation).

## Examples

standard input	standard output
5 10 3 3 4 2 2 5 2 4 3 3 1	Bunji 8
3 9 3 1 1 1 3 2 2 3 3 2	No winner
4 1 1 4	Ayumi 1

## Note

The first sample case corresponds to the following grid (with pieces numbered by the turn on which they were played).

	6	9		
	4	8	7	
10	3	1	2	5

Bunji was the first to meet the winning condition, with his first, third and fourth tokens forming a diagonal line of three. Note that Ayumi also met the winning condition on her next move, but this is immaterial.

The second sample case corresponds to the following grid (with pieces numbered by the turn on which they were played).

3	9	8
2	6	7
1	5	4

Neither player has  $k = 3$  in a row, so the game has **No winner**.

In the third sample case, Ayumi wins after playing a single piece, since  $k = 1$ .

# Reading Numbers

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

Theodore is learning to read numbers. He finds a number and reads out the digits, grouping runs of a single digit together. He then does the same for the resulting number, and so on.

For example, if the starting number is 1, then the sequence begins

1, 11, 21, 1211, 111221, 312211, 13112221, . . .

The fifth number is

  
The diagram shows the number 111221 with three orange brackets over the first three 1s labeled 'three 1s', two blue brackets over the next two 2s labeled 'two 2s', and one green bracket over the final 1 labeled 'one 1'.

so the sixth number is three-one-two-two-one-one, that is,

312211.

Given the starting number  $n$ , find the  $k$ th number that Theodore will read out.

## Input

The only line of input contains two integers  $n$  ( $1 \leq n \leq 1,000,000,000$ ) and  $k$  ( $1 \leq k \leq 50$ ), representing the starting number and the index required.

## Output

Output a single line, containing the  $k$ th number that Theodore reads out.

Note that this number could be very large.

## Examples

standard input	standard output
1 7	13112221
9 3	1119
22 43	22

## Note

In the third sample case, Theodore reads out 22 as two twos, so his second number is also 22. This continues forever.

# Clock Gallery

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

Mr Ritz has just opened his new hotel in Paris. Above the reception desk, he has mounted a vast gallery of  $n$  24-hour clocks to display the current time in many different cities, precise to the second. He proudly cuts the ribbon and welcomes the first guest to check in, only to be informed that the gallery is useless - he has forgotten to label the clocks, so guests have no idea which clock represents each city!

Worse still, Mr Ritz has been so busy preparing for the grand opening that he has not been outside in days, and has no idea of the current time in Paris, nor anywhere else in the world. All he knows about each city is its time difference from Paris, in equally precise detail.

Help Mr Ritz save face by labelling the clocks in a way that is consistent with the time differences.

## Input

The first line of input consists of a single integer  $n$  ( $1 \leq n \leq 1000$ ).

$n$  lines follow, the  $i$ th consisting of three two-digit *colon-separated* integers  $h_i$  ( $0 \leq h_i \leq 23$ ),  $m_i$  ( $0 \leq m_i \leq 59$ ) and  $s_i$  ( $0 \leq s_i \leq 59$ ), representing the hours, minutes and seconds displayed on the  $i$ th clock. No two of these lines are identical.

The last line of input consists of  $n$  space-separated integers  $d_1, \dots, d_n$  ( $-43199 \leq d_j \leq 43200$ ), the  $j$ th of these representing the time difference from Paris to the  $j$ th city in seconds. No two of the  $d_j$  are equal.

## Output

If there is no valid labelling of the clocks, output “Impossible” (without punctuation).

Otherwise, output a single line containing  $n$  space-separated integers, the city numbers corresponding to the clocks. There may be several correct solutions; any labelling of clocks which is consistent with the time differences will be accepted.

## Examples

standard input	standard output
4 06:00:00 04:00:00 10:00:00 01:00:00 7800 600 -10200 22200	1 2 4 3
3 11:59:59 12:00:00 12:00:02 1 0 -2	Impossible
2 00:00:00 23:59:59 0 1	2 1
2 00:00:00 12:00:00 0 43200	1 2

## Note

In the first sample input, the time in Paris is 03:50:00. Then the times in all cities are consistent:

- The time in city 1 (06:00:00) is 7800 seconds (2 hours 10 minutes) ahead of Paris time
- The time in city 2 (04:00:00) is 600 seconds (10 minutes) ahead of Paris time
- The time in city 3 (10:00:00) is 22200 seconds (6 hours 10 minutes) ahead of Paris time
- The time in city 4 (01:00:00) is 10200 seconds (2 hours 50 minutes) behind of Paris time

In the second sample input, the time differences from Paris ( $d_i$ ) are not consistent with the absolute times ( $h_i$ ).

In the third sample input, the time in Paris is 23:59:59. Then the times in all cities are consistent:

- The time in city 1 (00:00:00) is 1 second ahead of Paris time
- The time in city 2 (23:59:59) is in sync with Paris time

In the fourth sample input, the time in Paris is 00:00:00. Then the times in all cities are consistent:

- The time in city 1 (00:00:00) is in sync with Paris time
- The time in city 2 (12:00:00) is 43200 seconds (12 hours) ahead of Paris time

Note that for this sample input, there are multiple correct answers.

# Coin Puzzle

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

Juldys is exploring a dungeon. She has reached a mysterious room, where she finds  $n$  coins scattered on the floor, each with unique symbols on both sides. Juldys likes shiny things (and she particularly likes money) so she quickly picks up the coins. But as she collects the first coin, a heavy door slams down behind her, sealing her inside the room.

As she looks around, she sees runes glowing in red on the walls of the room, neatly organised into  $m$  rows each consisting of three runes. On closer inspection, she finds that each of the runes matches exactly one of the  $2n$  coin faces! Note that the same rune can appear in several different rows, and that some coin faces may not appear at all.

She also finds a locked door at the far end of the room, with  $n$  circular indentations that perfectly fit the coins. When she puts a coin into any of the indentations, all rows containing the rune facing outwards stop glowing, but when the coin is removed they glow once again.

Juldys figures out that she will have to place all  $n$  coins in a way that makes all the runes stop glowing, and only then will the door be unlocked. Help her find a way to place the coins, or determine that she is doomed.

## Input

The first line of input consists of two space-separated integers,  $n$  ( $3 \leq n \leq 22$ ) and  $m$  ( $0 \leq m \leq 100$ ), representing the number of coins and the number of rows of runes respectively.

$m$  lines follow, the  $i$ th of which contains three space-separated integers,  $r_{i,1}$ ,  $r_{i,2}$  and  $r_{i,3}$ , representing the runes in the  $i$ th row. Each  $r_{i,j}$  is a nonzero integer between  $-n$  and  $n$  inclusive, where a positive value  $k$  represents the “heads” side of coin  $k$  and a negative value  $-k$  represents the “tails” side of coin  $k$ . You are guaranteed that no two runes in any row refer to the same coin, i.e. that no two of the three values in any row are equal in absolute value.

## Output

If Juldys can unlock the door with some configuration of coins, output  $n$  space-separated letters, the  $i$ th of which is either H or T denoting which side of the  $i$ th coin should face up (heads or tails). There may be several correct solutions; any configuration of coins which stops all  $m$  rows glowing will be accepted.

If Juldys cannot unlock the door, output the word “Doomed” (without punctuation).









Examples
















standard input	standard output
3 5 1 2 3 1 -2 3 1 3 -2 -3 -1 2 1 2 3	H T T
3 8 3 1 2 3 -1 2 3 1 -2 3 -1 -2 2 1 -3 -2 1 -3 -1 2 -3 -1 -2 -3	Doomed


Note




For the sample cases, suppose the coins are as follows.




Coin	Heads	Tails
1		
2		
3		

In sample case 1, there are five rows of runes:

Row	Runes		
1			
2			
3			
4			
5			

Placing just the second coin showing tails will stop the second and third rows glowing, as these are the rows containing .

Placing the first coin showing heads and the other two showing tails will stop all rows glowing, as all rows contain at least one out of three displayed runes ,  and .

Note that this is not the only valid solution; if all three coins show heads, then ,  and  again account for all five rows.

In sample case 2, there are eight rows of runes, and no configuration of coins will account for all eight rows.