

Problem A. Birthday Bash (easy)

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

The only difference between easy and hard versions is the constraint on matrix dimensions.

It's April and Lawrence's birthday is just round the corner. Lawrence plans to throw a grandiose birthday party where he and his friends will vibe to classical pieces by Chopin, Mozart and Beethoven. Now, Lawrence will be hosting this party in the main hall of his house and wants your help to figure out the maximum number of friends he can invite. Lawrence wants all his friends to sit around one big rectangular table, therefore, he wants to know what is the largest rectangular table he can place such that maximum friends can sit around it.

The hall is represented as a 2D matrix of size $m \times n$ and the entries of the matrix are either '.' or 'X' indicating the particular cell is available or preoccupied by some furniture, respectively. You have to find the largest possible rectangle consisting of only free cells such that maximum number of friends can be accommodated.

Note: The number of friends that can sit around a table equals the perimeter of the table.

Constraints

- $1 \leq m, n \leq 100$

Input

The first line of input consists of two integers m and n respectively. Subsequently, there are m lines with n entries in each line which are either '.' or 'X' representing Lawrence's main hall.

Output

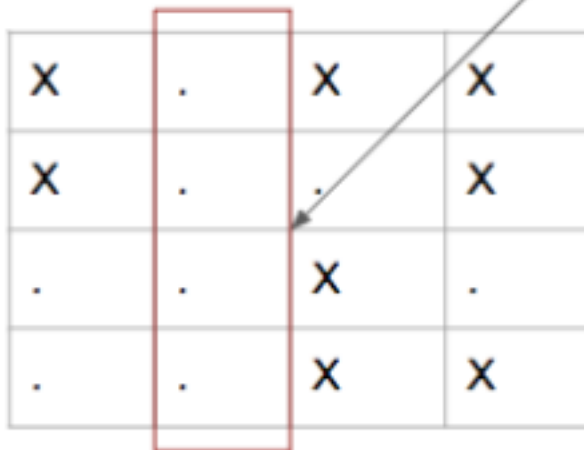
Print the maximum number of friends Lawrence can invite (note that you have to subtract 1 from the final answer to exclude Lawrence)

Example

standard input	standard output
4 4 X.XX X..X ..X. ..XX	9

Note

If table is placed along these empty cells, then length of table will be 4 units and breadth will be 1 unit. Hence, the seating capacity will be 10. Since, host will also occupy one seat, hence, he can invite 9 friends.



X	.	X	X
X	.	.	X
.	.	X	.
.	.	X	X

Problem B. Birthday Bash (hard)

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

The only difference between easy and hard versions is the constraint on matrix dimensions.

It's April and Lawrence's birthday is just round the corner. Lawrence plans to throw a grandiose birthday party where he and his friends will vibe to classical pieces by Chopin, Mozart and Beethoven. Now, Lawrence will be hosting this party in the main hall of his house and wants your help to figure out the maximum number of friends he can invite. Lawrence wants all his friends to sit around one big rectangular table, therefore, he wants to know what is the largest rectangular table he can place such that maximum friends can sit around it.

The hall is represented as a 2D matrix of size $m \times n$ and the entries of the matrix are either '.' or 'X' indicating the particular cell is available or preoccupied by some furniture, respectively. You have to find the largest possible rectangle consisting of only free cells such that maximum number of friends can be accommodated.

Note: The number of friends that can sit around a table equals the perimeter of the table.

Constraints

- $1 \leq m, n \leq 2500$

Input

The first line of input consists of two integers m and n respectively. Subsequently, there are m lines with n entries in each line which are either '.' or 'X' representing Lawrence's main hall.

Output

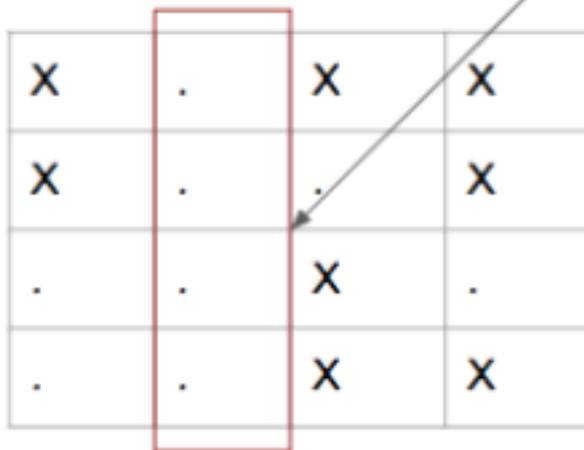
Print the maximum number of friends Lawrence can invite (note that you have to subtract 1 from the final answer to exclude Lawrence)

Example

standard input	standard output
4 4 X.XX X..X ..X. ..XX	9

Note

If table is placed along these empty cells, then length of table will be 4 units and breadth will be 1 unit. Hence, the seating capacity will be 10. Since, host will also occupy one seat, hence, he can invite 9 friends.



X	.	X	X
X	.	.	X
.	.	X	.
.	.	X	X

Problem C. Stacking the Queues

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

Kirdah is preparing for EDOBE's interview. He has asked Lawrence to take his mock interview. Lawrence was exploring some DSA problems which he could ask in the interview. He has surprisingly come up with a new problem of his own. The problem is as follows -

Let **S** be a stack. Each element in the stack **S** is going to be a queue. You can perform the following operations:

- **Push K** := meaning that you have to push a queue containing a single element with value **K** at its front.
- **Enqueue K** := meaning that you have to enqueue value **K** in the queue at the stack's top. If a stack is empty then perform **Push K** operation.
- **Pop** := meaning that you have to pop a stack element.
- **Dequeue** := meaning that you have to dequeue element from the queue at the stack's top and print that element. Note that if the queue becomes empty, then you must perform the **Pop** operation. If the stack is empty then print **-1**.
- **MaxSumPairs K** := meaning that in the queue present at the stack's top, print the maximum number of consecutive pairs with sum atleast **K**. After printing, perform the **Pop** operation.

He is given to perform **Q** of the above operations mentioned above. Kirdah was unable to solve the problem in the interview. Can you help him solve the problem?

Constraints:

- $1 \leq Q \leq 10^6$
- $1 \leq \text{size of stack } S, \text{ queues} \leq 10^6$
- $-10^9 \leq K \leq 10^9$

Input

The first line contains an integer **Q**, denoting the number of operations to be performed. Each of the following **Q** lines contains any one of the operations given.

Output

Print the outputs corresponding to the operation if any.

Examples

standard input	standard output
10 Push 2 Enqueue 1 Enqueue 1 MaxSumPairs 3 Dequeue Dequeue Dequeue Push 1 Pop Dequeue	1 -1 -1 -1 -1
6 Pop Dequeue Pop Dequeue Push 771666620 Pop	-1 -1

Problem D. Fire Spreading

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

Dee and Anny are working on a case where a building, consisting of $m \times n$ houses, caught fire in an accident. They have already determined which houses the fire started to spread from. As forensic investigators, they want to determine how long it took for the entire building to burn down, given that the fire can spread from one house to every house touching that house.

Constraints

- $1 \leq m, n \leq 10^4$
- $1 \leq m * n \leq 10^5$

Input

The first line of input consists of two integers m and n respectively. Subsequently, there are m lines with n entries in each line representing the binary 2D matrix. Where 1 represent the starting point of the fire spread and 0 represent the unburned houses.

Output

To determine the time it took for all the houses to burn down.

Examples

standard input	standard output
3 3 0 0 0 0 1 0 0 0 0	1
3 4 1 0 0 0 0 0 0 0 0 0 0 1	2
4 4 1 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0	1

Problem E. Card Game

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

Rohu and Ishu are playing a card game. They have a deck of n cards, each with some value. In each round of the game, they perform the following steps:

1. They draw K cards from the top of the deck.
2. They note the highest value among the drawn cards.
3. They remove the first card from the drawn set.
4. Steps 1-3 are repeated until there are fewer than K cards left in the deck.

Once the game has ended, the winner is determined as follows-

The total score for the Player is calculated as the sum of the highest values noted by him during the game.

The player with the highest total score is declared the winner. If both players have the same total score, the game is declared a draw.

Note: The game Always starts with Ishu

Input

The first line of input consists of two integers n and k where n is The number of cards in the deck ($1 \leq n \leq 10^5$) and K is The number of cards drawn in each round ($1 \leq K \leq n$) and the second line contains n space-separated numbers a_1, a_2, \dots, a_n , where a_i is the value of the i -th card in the deck ($1 \leq a_i \leq 10^9$)

Output

Print "Rohu" if Rohu wins, "Ishu" if Ishu wins, or "draw" if the scores are equal.

Examples

standard input	standard output
6 3 4 6 2 8 1 7	rohu
10 2 3 7 1 9 4 8 2 6 5 10	ishu
6 3 2 6 3 1 9 7	draw

Note

In Testcase 1:

There are 6 cards in the deck and 3 cards are drawn in each round. The values of the cards in the deck are [4, 6, 2, 8, 1, 7].

Ishu draws the first 3 cards: [4, 6, 2]. The highest value is 6. Ishu removes the first card from the drawn set: [4, 6, 2].

Rohu draws the next 3 cards: [6, 2, 8, 1, 7]. The highest value is 8. Rohu removes the first card from the drawn set: [6, 2, 8].

Ishu draws the next 3 cards: [2, 8, 1, 7]. The highest value is 8. Ishu removes the first card from the drawn set: [2, 8, 1].

Rohu draws the first 3 cards: [8, 1, 7]. The highest value is 8. Rohu removes the first card from the drawn set: [8, 1, 7].

The game ends because there are only 2 cards left in the deck.

Score calculation:

Ishu's score: $6 + 8 = 14$

Rohu's score: $8 + 8 = 16$

Since Rohu's score is higher than Ishu's score, the winner is Rohu. Therefore, the output is "Rohu".