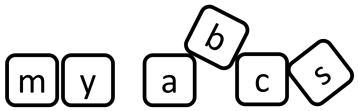
Problem A — limit 1 second

Alphabet



A string of lowercase letters is called *alphabetical* if deleting zero or more of its letters can result in the *alphabet string* "`abcdefghijklmnopqrstuvwxyz`".

Given a string s, determine the minimum number of letters to insert anywhere in the string to make it alphabetical.

Input

The input consists of a single line containing the string s ($1 \leq |s| \leq 50$).

It is guaranteed that s consists of lowercase ASCII letters '`a`' to '`z`' only.

Output

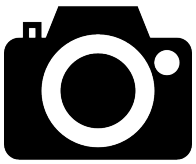
Print, on a single line, a single integer indicating the minimum number of letters that must be inserted in order to make the string s alphabetical.

Sample Input	**Sample Output**
`xyzabcdefghijklmnopqrstuvw`	`3`

Sample Input	**Sample Output**
`aiemckgobjfndlhp`	`20`

Problem **B** — limit 1 second

Cameras



Your street has n houses, conveniently numbered from 1 to n. Out of these n houses, k of them have security cameras installed. Mindful of gaps in coverage, the Neighborhood Watch would like to ensure that every set of r consecutive houses has at least two different houses with cameras. What is the minimum number of additional cameras necessary to achieve this?

Input

The first line of input contains three integers, n ($2 \leq n \leq 100{,}000$), k ($0 \leq k \leq n$), and r ($2 \leq r \leq n$).

The next k lines of input contain the distinct locations of the existing cameras.

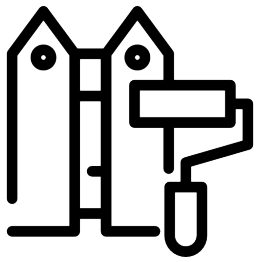
Output

Print, on a single line, a single integer indicating the minimum number of cameras that need to be added.

Sample Input	**Sample Output**
15 5 4 2 5 7 10 13	3

Problem **C** — limit 4 seconds

Paint



You are painting a fence with n sections, numbered from 1 to n. There are k artists, each willing to paint their design on a specific portion of the fence. However, artists will never agree to have their section painted over, so they will only paint their portion of the fence if no one else will paint any part of it.

You want to select a set of painters that does not conflict to minimize the number of unpainted sections.

Input

The first line contains two positive integers n ($1 \leq n \leq 10^{18}$) and k ($1 \leq k \leq 200{,}000$).

Each of the next k lines contains two positive integers a_i and b_i, where $1 \leq a_i \leq b_i \leq n$, indicating that the ith artist wants to paint all sections between section a_i and section b_i, inclusive.

Output

Print, on a single line, a single integer indicating the minimum number of unpainted sections.

Sample Input	**Sample Output**
8 3 1 3 2 6 5 8	1

Problem **D** — limit 1 second

Tournament Wins

You are one of 2^k competitors invited to enter a single elimination tournament. You are ranked rth in the published rankings. Furthermore, you know that in any match between two players, the one ranked higher will always win.

The only source of uncertainty is the bracket. If every possible tournament bracket is equally likely, determine your expected number of wins in the tournament. Your expected number of wins is the average number of your wins over all possible tournament bracket orderings.

Input

The input consists of a single line containing the two space-separated integers k ($1 \leq k \leq 20$) and r ($1 \leq r \leq 2^k$).

Output

Print, on a single line, your expected number of wins in the tournament, rounded and displayed to exactly five decimal places. The sixth digit after the decimal point of the exact answer will never be 4 or 5 (eliminating complex rounding considerations).

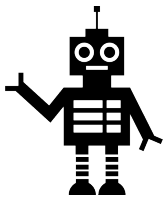
Be careful about very small or very large numbers during intermediate steps.

Sample Input	**Sample Output**
`3 3`	`1.00000`

Sample Input	**Sample Output**
`20 130`	`11.65203`

Problem **E** — limit 1 second

Buggy Robot



You are trying to program a robot to navigate through a 2-dimensional maze and find the exit.

The maze can be represented as a grid with n rows and m columns. Some grid cells have obstacles that the robot cannot pass. The other cells are empty, which the robot can freely pass. Exactly one of the empty cells in the grid is marked as the exit, and the robot will exit the maze immediately once it reaches there.

You can program the robot by sending it a *command string*. A command string consists of characters '`L`', '`U`', '`R`', '`D`', corresponding to the directions left, up, right, down, respectively. The robot will then start executing the commands, by moving to an adjacent cell in the directions specified by the command string. If the robot would run into an obstacle or off the edge of the grid, it will ignore the command, but it will continue on to remaining commands. The robot will also ignore all commands after reaching the exit cell.

Your friend sent you a draft of a command string, but you quickly realize that the command string will not necessarily take the robot to the exit. You would like to fix the string so that the robot will reach the exit square. In one second, you can delete an arbitrary character, or add an arbitrary character at an arbitrary position. Find how quickly you can fix your friend's command string.

You do not care how long it takes the robot to find the exit, but only how long it takes to repair the command string.

Input

The first line of input contains the two integers n and m ($1 \leq n, m \leq 50$).

Each of the next n lines contains m characters, describing the corresponding row of the grid. Empty cells are denoted as '`.`', the robot's initial position is denoted as '`R`', obstacles are denoted as '`#`', and the exit is denoted as '`E`'.

The next and final line of input contains your friend's command string, consisting of between 1 and 50 characters, inclusive.

It is guaranteed that the grid contains exactly one '`R`' and one '`E`', and that there is always a path from '`R`' to '`E`'.

Output

Print, on a single line, a single integer indicating the minimum amount of time to fix the program.

Sample Input	**Sample Output**
3 3 R.. .#. ..E LRDD	1

Sample Input	**Sample Output**
2 4 R.#. #..E RRUUDDRRUUUU	0

Problem **F** — limit 3 seconds

Windy Path

There are n obstacles placed in a field. Your task is to design a course that visits each obstacle exactly once, in any order, following a straight line between consecutive obstacles, without ever crossing itself.

The catch? The sequence of turn directions (left or right) has already been decided, in a string of length $n-2$. If the ith character of the turn sequence is '`L`', then the locations of the ith, $(i+1)$th, and $(i+2)$th obstacles, in that order, must form a counterclockwise angle. If it is '`R`', they must form a clockwise angle.

Input

The first line of input contains a single integer n ($3 \leq n \leq 50$).

Each of the next n lines contains two space-separated integers x_i and y_i ($1 \leq x_i, y_i \leq 1{,}000$), giving the coordinates of obstacle i.

The next and final line will contain a single string with exactly $n-2$ characters consisting of only '`L`' and '`R`', representing the sequence of turn directions.

It is guaranteed that no three obstacles will be collinear.

Output

If no solution is possible, print, on a single line, the integer '`-1`'. Otherwise, print, on a single line, any permutation of the obstacles that satisfies the requirements. The permutation should be given as n distinct space-separated integers p_i with $1 \leq p_i \leq n$, and this ordering of the points should satisfy the turn directions indicated by the turn sequence.

If there are multiple possible solutions, print any of them.

Sample Input	Sample Output
4 2 2 2 1 1 2 1 1 LR	1 3 2 4