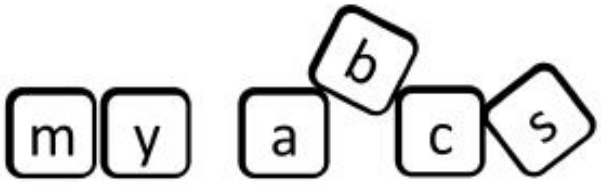
Problem A. Alphabet

Source file name:	Alphabet.c, Alphabet.cpp, Alphabet.java, Alphabet.py
Input:	`Standard`
Output:	`Standard`



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

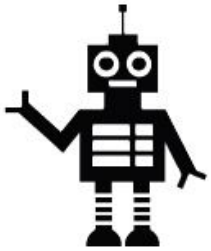
Print, on a single line, a single integer indicating the minimum number of letters that must be inserted in order to make the string s alphabetical.

Example

Input	Output
`xyzabcdefghijklmnopqrstuvw`	3
`aiemckgobjfndlhp`	20

Problem B. Buggy Robot

Source file name:	Buggy.c, Buggy.cpp, Buggy.java, Buggy.py
Input:	`Standard`
Output:	`Standard`



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.

Example

Input	Output
3 3 R.. .#. ..E LRDD	1
2 4 R.#. #..E RRUUDDRRUUUU	0

Problem C. Cameras

Source file name:	Cameras.c, Cameras.cpp, Cameras.java, Cameras.py
Input:	`Standard`
Output:	`Standard`

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 ($2 \leq n \leq 10^5$), 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.

Example

Input	Output
15 5 4 2 5 7 10 13	3

Problem D. Contest Strategy

Source file name:	Conteststrat.c, Conteststrat.cpp, Conteststrat.java, Conteststrat.py
Input:	`Standard`
Output:	`Standard`

You are participating in the Association for Computing Machinery's Intercollegiate Programming Competition (ACM ICPC). You must complete a set of n problems. Since you are an experienced problem solver, you can read a problem and accurately estimate how long it will take to solve it, in a negligible amount of time.

Let t_i be the time it will take to solve the ith problem. Your strategy for the contest is as follows:

1. Read k random problems.
2. Choose a problem that you have read that will take the shortest time to solve (if there are ties, choose any of them arbitrarily).
3. Solve the problem, and read a random unread problem (if there is any).
4. If there are still unsolved problems, go back step 2.

Your penalty time for the contest is defined by the sum of submission times for all the problems. Of course, your penalty time depends on the order in which the problems are read. What is the sum of penalty times, over all $n!$ possible different orders you read the problems in? Since the result can be very large, find the answer modulo $10^9 + 7$.

Input

The first line of input contains two space-separated integers n and k ($1 \leq k \leq n \leq 300$).

The ith line of the next n lines contains a single integer t_i ($1 \leq t_i \leq 10^6$).

Output

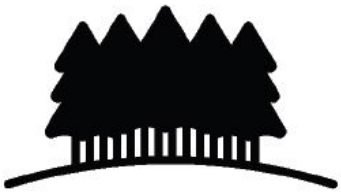
Print, on a single line, a single integer representing the sum of penalty times over all possible orders you read the problems in, modulo $10^9 + 7$.

Example

Input	Output
4 3 1 3 2 1	336
10 2 1000000 2 152 49 93064 438953 438 9238 9065 1274	513850896

Problem E. Enclosure

Source file name:	Enclosure.c, Enclosure.cpp, Enclosure.java, Enclosure.py
Input:	`Standard`
Output:	`Standard`



In the Dark Forest, the territory you control is defined by the smallest convex polygon that contains all trees you control. Your power is defined by the area of the territory you control.

You currently control k out of n trees in the Dark Forest. What is the highest power you can achieve by gaining control over a single additional tree somewhere in the forest?

Input

The first line of input consists of two space-separated integers n and k ($3 \leq k < n \leq 10^5$).

Next follow n lines each with two space-separated integers x_i and y_i ($|x_i|, |y_i| \leq 10^9$) specifying the locations of the n trees. You control the first k trees given in the list; the other $n - k$ trees do not belong to you. (Note that some of these may still be inside your territory.)

It is guaranteed that no three trees have collinear locations.

Output

Print, on a single line, the maximum power you can achieve by gaining control over a single additional tree. The output should be rounded and displayed to exactly one decimal place.

Example

Input	Output
5 3 -5 -5 -5 5 5 -5 -4 6 5 5	100.0

Problem F. Illumination

Source file name:	Illumination.c, Illumination.cpp, Illumination.java, Illumination.py
Input:	`Standard`
Output:	`Standard`

You inherited a haunted house. Its floor plan is an n-by-n square grid with l lamps in fixed locations and no interior walls. Each lamp can either illuminate its row or its column, but not both simultaneously. The illumination of each lamp extends by r squares in both directions, so a lamp unobstructed by an exterior wall of the house can illuminate as many as $2r + 1$ squares.

If a square is illuminated by more than one lamp in its row, or by more than one lamp in its column, the resulting bright spot will scare away ghosts forever, diminishing the value of your property. Is it possible for all lamps to illuminate a row or column, without scaring any ghosts? Note that a square illuminated by two lamps, one in its row and the other in its column, will not scare away the ghosts.

Input

The first line of input contains three positive integers, n, r and l ($1 \leq n, r, l \leq 10^3$).

Each of the next l lines contains two positive integers r_i and c_i ($1 \leq r_i, c_i \leq n$), indicating that there is a lamp in row r_i and column c_i.

It is guaranteed that all lamps are in distinct locations.

Output

Print, on a single line, `YES` if it is possible to illuminate all lamps as stated above; otherwise, print `NO`.

Example

Input	Output
3 2 5 1 1 1 3 3 1 3 3 2 2	YES
3 2 6 1 1 1 2 1 3 3 1 3 2 3 3	NO

Problem G. Maximum Island

Source file name:	Maximumisl.c, Maximumisl.cpp, Maximumisl.java, Maximumisl.py
Input:	`Standard`
Output:	`Standard`

You are mapping a faraway planet using a satellite.

Your satellite has captured an image of the planet's surface. The photographed section can be modeled as a grid. Each grid cell is either land, water, or covered by clouds. Clouds mean that the surface could either be land or water, but we can't tell.

An island is a set of connected land cells. Two cells are considered connected if they share an edge.

Given the image, determine the maximum number of islands that is consistent with the given information.

Input

The first line of input contains two space-separated integers n and m ($1 \leq n, m \leq 40$).

Each of the next n lines contains m characters, describing the satellite image. Land cells are denoted by '`L`', water cells are denoted by '`W`', and cells covered by clouds are denoted by '`C`'.

Output

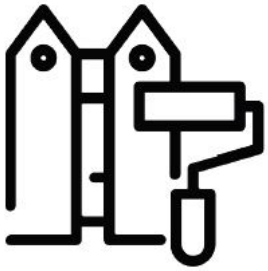
Print, on a single line, a single integer indicating the maximum number of islands that is consistent with the given grid.

Example

Input	Output
5 4 LLWL CCCC CCCC CCCC LWLL	8

Problem H. Paint

Source file name: Paint.c, Paint.cpp, Paint.java, Paint.py
Input: Standard
Output: Standard



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 ($1 \leq k \leq 2 * 10^5$).

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.

Example

Input	Output
8 3 1 3 2 6 5 8	1

Problem I. Postman

Source file name:	Postman.c, Postman.cpp, Postman.java, Postman.py
Input:	`Standard`
Output:	`Standard`

A postman delivers letters to his neighbors in a one-dimensional world.

The post office, which contains all of the letters to begin with, is located at $x = 0$, and there are n houses to which the postman needs to deliver the letters. House i is located at position x_i , and there are m_i letters that need to be delivered to this location. But the postman can only carry k letters at once.

The postman must start at the post office, pick up some number of letters less than or equal to his carrying capacity, and then travel to some of the houses dropping off letters. He must then return to the post office, repeating this process until all letters are delivered. At the end he must return to the post office.

The postman can travel one unit of distance in one unit of time.

What is the minimum amount of time it will take the postman to start at the post office, deliver all the letters, and return to the post office?

Input

The first line of input contains two space-separated integers n ($1 \leq n \leq 10^3$) and k ($1 \leq k \leq 10^7$).

Each of the next n lines contains two space-separated integers x_i ($|xi| \leq 10^7$) and m_i ($1 \leq m_i \leq 10^7$).

Output

Print, on a single line, the minimum amount of time it will take to complete the mail delivery route.

Example

Input	Output
4 10 -7 5 -2 3 5 7 9 5	42
7 1 9400000 10000000 9500000 10000000 9600000 10000000 9700000 10000000 9800000 10000000 9900000 10000000 10000000 10000000	1358000000000000

Problem J. Shopping

Source file name:	Shopping.c, Shopping.cpp, Shopping.java, Shopping.py
Input:	`Standard`
Output:	`Standard`

The sale bin of Big Box Bargains contains n products in a row. The ith item has price a_i per unit. There is no limit to the quantity of any item.

There are q customers who will enter the store to buy items. The ith customer has v_i dollars, starts at item l_i and walks to the right to item r_i (inclusive), one item at a time.

Each time they encounter an item, they will buy as many units of the item as they can afford.

You are now wondering, for each customer, how much money they will have left after buying items.

Input

The first line of input contains two space-separated integers n and q ($1 \leq n, q \leq 2 * 10^5$).

The next line of input contains n space-separated integers a_i ($1 \leq a_i \leq 10^{18}$).

Each of the next q lines contains three space-separated integers v_i ($1 \leq v_i \leq 10^{18}$), l_i, and r_i ($1 \leq l_i \leq r_i \leq n$).

Output

For each of the q customers, print, on a single line, a single integer indicating the remaining amount of money after shopping.

Example

Input	Output
5 3 5 3 2 4 6 8 5 5 107 1 4 7 3 5	2 0 1

Problem K. Tournament Wins

Source file name:	Tournament.c, Tournament.cpp, Tournament.java, Tournament.py
Input:	`Standard`
Output:	`Standard`

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.

Example

Input	Output
3 3	1.00000
20 130	11.65203

Problem L. Barbells

Source file name:	Barbells.c, Barbells.cpp, Barbells.java, Barbells.py
Input:	`Standard`
Output:	`Standard`

Your local gym has n barbells and m plates. In order to prepare a weight for lifting, you must choose a single barbell, which has two sides. You then load each side with a (possibly empty) set of plates. For safety reasons, the plates on each side must sum to the same weight. What weights are available for lifting?

For example, suppose that there are two barbells weighing 100 and 110 grams, and five plates weighting 1, 4, 5, 5, and 6 grams, respectively. Then, there are six possible weights available for lifting. The table below shows one way to attain the different weights:

Barbell	Left side	Right side	Total
100	0	0	100
100	5	5	110
100	$1+5$	6	112
110	5	5	120
110	$1+5$	6	122
110	$5+5$	$4+6$	130

Input

The first line of input contains the space-separated integers n and m ($1 \leq n, m \leq 14$).

The second line of input contains n space-separated integers $b_1, ..., b_n$ ($1 \leq b_i \leq 10^8$), denoting the weights of the barbells in grams.

The third line of input contains m space-separated integers $p_1, ..., p_m$ ($1 \leq p_i \leq 10^8$), denoting the weights of the plates in grams.

Output

Print a sorted list of all possible distinct weights in grams, one per line.

Example

Input	Output
2 5 100 110 5 5 1 4 6	100 110 112 120 122 130