Problem A. Jumbled Compass

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second

Jonas is developing the JUxtaPhone and is tasked with animating the compass needle. The API is simple: the compass needle is currently in some direction (between 0 and 359 degrees, with north being 0, east being 90), and is being animated by giving the degrees to spin it. If the needle is pointing north, and you give the compass an input of 90, it will spin clockwise (positive numbers mean clockwise direction) to stop at east, whereas an input of -45 would spin it counterclockwise to stop at north west.

The compass gives the current direction the phone is pointing and Jonas' task is to animate the needle taking the *shortest path* from the current needle direction to the correct direction. Many `if`s, moduli, and even an arctan later, he is still not convinced his `minimumDistance` function is correct; he calls you on the phone.

Input

The first line of input contains an integer $n_1 (0 \leq n_1 \leq 359)$, the current direction of the needle. The second line of input contains an integer $n_2 (0 \leq n_2 \leq 359)$, the correct direction of the needle.

Output

Output the change in direction that would make the needle spin the shortest distance from n_1 to n_2. A positive change indicates spinning the needle clockwise, and a negative change indicates spinning the needle counter-clockwise. If the two input numbers are diametrically opposed, the needle should travel clockwise. I.e., in this case, output 180 rather than -180.

Examples

stdin	stdout
315 45	90
180 270	90
45 270	-135

Problem B. Game Rank

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second

The gaming company Sandstorm is developing an online two player game. You have been asked to implement the ranking system. All players have a rank determining their playing strength which gets updated after every game played. There are 25 regular ranks, and an extra rank, "`Legend`", above that. The ranks are numbered in decreasing order, 25 being the lowest rank, 1 the second highest rank, and `Legend` the highest rank.

Each rank has a certain number of "stars" that one needs to gain before advancing to the next rank. If a player wins a game, she gains a star. If before the game the player was on rank 6—25, and this was the third or more consecutive win, she gains an additional bonus star for that win. When she has all the stars for her rank (see list below) and gains another star, she will instead gain one rank and have one star on the new rank.

For instance, if before a winning game the player had all the stars on her current rank, she will after the game have gained one rank and have 1 or 2 stars (depending on whether she got a bonus star) on the new rank. If on the other hand she had all stars except one on a rank, and won a game that also gave her a bonus star, she would gain one rank and have 1 star on the new rank.

If a player on rank 1—20 loses a game, she loses a star. If a player has zero stars on a rank and loses a star, she will lose a rank and have all stars minus one on the rank below. However, one can never drop below rank 20 (losing a game at rank 20 with no stars will have no effect).

If a player reaches the `Legend` rank, she will stay legend no matter how many losses she incurs afterwards.

The number of stars on each rank are as follows:

- Rank 25—21: 2 stars
- Rank 20—16: 3 stars
- Rank 15—11: 4 stars
- Rank 10—1: 5 stars

A player starts at rank 25 with no stars. Given the match history of a player, what is her rank at the end of the sequence of matches?

Input

The input consists of a single line describing the sequence of matches. Each character corresponds to one game; '`W`' represents a win and '`L`' a loss. The length of the line is between 1 and $10,000$ characters (inclusive).

Output

Output a single line containing a rank after having played the given sequence of games; either an integer between 1 and 25 or "`Legend`".

Examples

stdin	stdout
WW	25
WWW	24
WWWW	23
WLWLWLWL	24
WWWWWWWWWLLWW	19
WWWWWWWWWLWWL	18

Problem C. Islands

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds

You are mapping a faraway planet using a satellite. The planet's surface can be modeled as a grid. The satellite has captured an image of the surface. Each grid square is either land (denoted as '`L`'), water (denoted as '`W`'), or covered by clouds (denoted as '`C`'). Clouds mean that the surface could either be land or water; you cannot tell.

An island is a region of land where every grid cell in the island is connected to every other by some path, and every leg of the path only goes up, down, left or right. Given an image, determine the minimum number of islands that is consistent with the given image.

Input

The first line of input contains two integers, r and c ($1 \leq r, c \leq 50$), which are the number of rows and the number of columns of the image. The next r lines will each contain exactly c characters, consisting only of '`L`' (representing `Land`), '`W`' (representing `Water`), and '`C`' (representing `Clouds`).

Output

Output a single integer, which is the minimum number of islands possible.

Examples

stdin	stdout
4 5 CCCCC CCCCC CCCCC CCCCC	0
3 2 LW CC WL	1

Problem D. Bless You Autocorrect!

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds

Typing on phones can be tedious. It is easy to make typing mistakes, which is why most phones come with an autocorrect feature. Autocorrect not only fixes common typos, but also suggests how to finish the word while you type it. Jenny has recently been pondering how she can use this feature to her advantage, so that she can send a particular message with the minimum amount of typing.

The autocorrect feature on Jenny's phone works like this: the phone has an internal dictionary of words sorted by their frequency in the English language. Whenever a word is being typed, autocorrect suggests the most common word (if any) starting with all the letters typed so far. By pressing tab, the word being typed is completed with the autocorrect suggestion. Autocorrect can only be used after the first character of a word has been typed — it is not possible to press tab before having typed anything. If no dictionary word starts with the letters typed so far, pressing tab has no effect.

Jenny has recently noticed that it is sometimes possible to use autocorrect to her advantage even when it is not suggesting the correct word, by deleting the end of the autocorrected word. For instance, to type the word "`autocorrelation`", Jenny starts typing "`aut`", which then autocorrects to "`autocorrect`" (because it is such a common word these days!) when pressing tab. By deleting the last two characters ("`ct`") and then typing the six letters "`lation`", the whole word can be typed using only 3 ("`aut`") + 1 (tab) + 2 (backspace twice) + 6 ("`lation`") = 12 keystrokes, 3 fewer than typing "`autocorrelation`" without using autocorrect.

Given the dictionary on the phone and the words Jenny wants to type, output the minimum number of keystrokes required to type each word. The only keys Jenny can use are the letter keys, tab and backspace.

Input

The first line of input contains two positive integers n ($1 \leq n \leq 10^5$), the number of words in the dictionary, and m ($1 \leq m \leq 10^5$), the number of words to type. Then follow n lines with one word per line, sorted in decreasing order of how common the word is (the first word is the most common). No word appears twice in the dictionary. Then follow m lines, containing the words to type.

The dictionary and the words to type only use lower case letters '`a`'-'`z`'. The total size of the input file is at most **1 MB**.

Output

For each word to type, output a line containing the minimum number of keystrokes required to type the corresponding word.

Examples

stdin	stdout
5 5 austria autocorrect program programming computer autocorrelation programming competition zyx austria	12 4 11 3 2
5 3 yogurt you blessing auto correct bless you autocorrect	5 3 9

Problem E. Paint

Input file: stdin
Output file: stdout
Time limit: 2 seconds

You are painting a picket fence with n slats, numbered from 1 to n. There are k painters willing to paint a specific portion of the fence. However, they don't like each other, and each painter will only paint their given portion of the fence if no other painter overlaps their portion.

You want to select a subset of painters that do not conflict with each other, in order to minimize the number of unpainted slats. For example, suppose there are 8 slats, and 3 painters. One painter wants to paint slats $1 \to 3$, one wants to paint $2 \to 6$, and one wants to paint $5 \to 8$. By choosing the first and last painters, you can paint most of the slats, leaving only a single slat (slat 4) unpainted, with no overlap between painters.

Input

The first line of input contains two integers n ($1 \leq n \leq 10^{18}$) and k ($1 \leq k \leq 200,000$), where n is the number of slats and k is the number of painters. Each of the next k lines contains two integers a and b ($1 \leq a \leq b \leq n$), indicating that this painter wants to paint all of the slats between a and b, inclusive.

Output

Output a single integer, which is the smallest number of slats that go unpainted with an optimal selection of painters

Examples

stdin	stdout
8 3 1 3 2 6 5 8	1

Problem F. Exponial

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds

Everybody loves big numbers (if you do not, you might want to stop reading at this point). There are many ways of constructing really big numbers known to humankind, for instance:

- Exponentiation: $42^{2016} = 42 \cdot 42 \cdot \ldots \cdot 42$, 2016 times.
- Factorials: $2016! = 2016 \cdot 2015 \cdot \ldots \cdot 2 \cdot 1$.

In this problem we look at their lesser-known love-child the *exponial*, which is an operation defined for all positive integers n as

$$\text{exponial}(n) = n^{(n-1)^{(n-2)^{\cdots^{2^1}}}}$$

For example, $\text{exponial}(1) = 1$ and $\text{exponial}(5) = 5^{4^{3^{2^1}}} = 6.206 \cdot 10^{183230}$ which is already pretty big. Note that exponentiation is right-associative: $a^{b^c} = a^{(b^c)}$.

Since the exponials are really big, they can be a bit unwieldy to work with. Therefore we would like you to write a program which computes $\text{exponial}(n)$ `mod` m (the remainder of $\text{exponial}(n)$ when dividing by m).

Input

The input consists of two integers n ($1 \leq n \leq 10^9$) and m ($1 \leq m \leq 10^9$).

Output

Output a single integer, the value of $\text{exponial}(n)$ `mod` m.

Examples

stdin	stdout
2 42	2
5 123456789	16317634
94 265	39

Problem G. Artwork

Input file:	`stdin`
Output file:	`stdout`
Time limit:	5 seconds

A template for an artwork is a white grid of $n \times m$ squares. The artwork will be created by painting q horizontal and vertical black strokes. A stroke starts from square (x_1, y_1), ends at square (x_2, y_2) ($x_1 = x_2$ or $y_1 = y_2$) and changes the color of all squares (x, y) to black where $x_1 \leq x \leq x_2$ and $y_1 \leq y \leq y_2$.

The beauty of an artwork is the number of regions in the grid. Each region consists of one or more white squares that are connected to each other using a path of white squares in the grid, walking horizontally or vertically but not diagonally. The initial beauty of the artwork is 1. Your task is to calculate the beauty after each new stroke. The following figure illustrates how the beauty of the artwork varies in Sample Input 1.

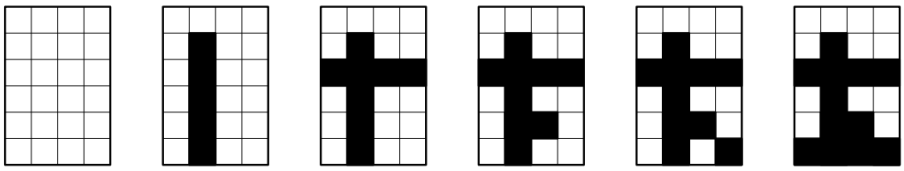

Figure A.1: Illustration of Sample Input 1.

Input

The first line of input contains three integers n, m and q ($1 \leq n, m \leq 1000, 1 \leq q \leq 10^4$). Then follow q lines that describe the strokes. Each line consists of four integers $x1, y1, x2$ and $y2$ ($1 \leq x_1 \leq x_2 \leq n, 1 \leq y_1 \leq y_2 \leq m$). Either $x_1 = x_2$ or $y_1 = y_2$ (or both).

Output

For each of the q strokes, output a line containing the beauty of the artwork after the stroke.

Examples

stdin	stdout
4 6 5 2 2 2 6 1 3 4 3 2 5 3 5 4 6 4 6 1 6 4 6	1 3 3 4 3

Problem H. Card Hand Sorting

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second

When dealt cards in the card game Plump it is a good idea to start by sorting the cards in hand by suit and rank. The different suits should be grouped and the ranks should be sorted within each suit. But the order of the suits does not matter and within each suit, the cards may be sorted in either ascending or descending order on rank. It is allowed for some suits to be sorted in ascending order and others in descending order.

Sorting is done by moving one card at a time from its current position to a new position in the hand, at the start, end, or in between two adjacent cards. What is the smallest number of moves required to sort a given hand of cards?

Input

The first line of input contains an integer n ($1 \leq n \leq 52$), the number of cards in the hand. The second line contains n pairwise distinct space-separated cards, each represented by two characters. The first character of a card represents the rank and is either a digit from 2 to 9 or one of the letters `T`, `J`, `Q`, `K`, and `A` representing `Ten`, `Jack`, `Queen`, `King` and `Ace`, respectively, given here in increasing order. The second character of a card is from the set `{s, h, d, c}` representing the suits spades, hearts, diamonds, and clubs.

Output

Output the minimum number of card moves required to sort the hand as described above.

Examples

stdin	stdout
4 2h Th 8c Qh	1
7 9d As 2s Qd 2c Jd 8h	2
4 2h 3h 9c 8c	0

Problem I. Illumination

Input file:	`stdin`
Output file:	`stdout`
Time limit:	5 seconds

Consider a square grid with lamps in fixed locations. Each lamp can either illuminate its row or its column, but not both. The illumination of each lamp extends over a limited distance.

Any square in the grid should only be illuminated by at most one lamp in its row and by at most one lamp in its column (one of each is acceptable, as is just the row, just the column, or neither). Determine if it is possible for all lamps to be lit while satisfying these constraints.

Input

The first line of input contains three positive integers, n, r and k ($1 \leq n, r, k \leq 1,000, k \leq n \times n$), where n is the size of one side of the square grid, r is the maximum reach of a lamp, and k is the number of lamps. The next k lines will each contain two positive integers i and j ($1 \leq i, j \leq n$), indicating that there is a lamp in the grid at row i, column j.

Each lamp can illuminate squares at most r units away, and can also illuminate its own square, so the maximum number of squares it can illuminate is $2r + 1$. All lamps will be in distinct locations.

Output

Output a single integer, 1 if it is possible to light all of the lamps and 0 if it is not possible.

Examples

stdin	stdout
3 2 5 1 1 1 3 3 1 3 3 2 2	1
3 2 6 1 1 1 2 1 3 3 1 3 2 3 3	0

Problem J. InTents

Input file:	stdin
Output file:	stdout
Time limit:	5 seconds

A circus is constructing their tent. The tent is a large piece of canvas held up by a given number of various length tent poles. The tent poles go in specific places on the ground, but which tent pole goes in which place is up to you. You need to choose a placement for the given tent poles that maximizes the total volume under the tent.

There will always be one central pole at the origin; the other poles are distributed around the periphery. The tent is always drawn tight between the central pole and two adjacent poles on the periphery, forming a perfect triangle. Only the volume under these triangles formed by two adjacent outer poles and the central origin pole counts towards the total volume. Adjacency is by angle around the origin.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. The first line of input contains an integer n ($3 \leq n \leq 30$), which is the number of poles.

The next $n-1$ lines each contains two integers x and y ($-1,000 \leq x, y \leq 1,000$), representing a 2D coordinate, giving the locations where the poles may be placed. The locations may not be in order around the origin. After that, there will be n lines, each containing a single integer h ($1 \leq h \leq 100$). These are the heights of the poles.

One pole must be placed at the origin, and the rest must be placed at the (x, y) coordinates in the input. The (x, y) locations will surround the origin; that is, the polygon formed by the (x, y) locations, in order (by angle around the origin), will strictly include the origin. No two holes will be at the same angle with the origin (i.e. no triangle of roof fabric will have area 0).

Output

Output a single floating point number, which is the maximum volume achievable under the tent. Output this number to exactly two decimal places, rounded.

Examples

stdin	stdout
5 100 100 -200 -200 300 -300 -400 400 30 20 50 60 10	8566666.67

Problem K. Water

Input file:	`stdin`
Output file:	`stdout`
Time limit:	5 seconds

A water company is trying to provide water from its pumping station to a mansion. The company owns n water stations, numbered $1 \ldots n$, which are connected by a variety of pipes. Water can flow through both directions of a pipe, but the total amount of water that can flow through the pipe is bounded by the capacity of the pipe.

The water company is constantly improving the pipes, increasing the capacity of various pipes. The water company is conducting k improvements (each of which is permanent after it is executed). An improvement consists of taking a pipe between two locations and increasing its capacity by a fixed amount, or installing a pipe between two locations which are not directly connected by a pipe.

After each improvement, the water company wants to know the maximum amount of water the mansion could receive.

Input

The first line of input contains three integers, n ($2 \leq n \leq 100$), p ($0 \leq p \leq \frac{n(n-1)}{2}$), and k ($1 \leq k \leq 10,000$), where n is the number of stations, p is the number of initial pipes, and k is the number of improvements. The first station in the list is always the pumping station, and the second is always the mansion.

The next p lines will describe the pipes in the initial setup. The lines will each contain three integers, a, b ($1 \leq a < b \leq n$) and c ($1 \leq c \leq 1,000$), which indicates that stations a and b are connected by a pipe with capacity c. No (a, b) pair will appear more than once in this section.

The next k lines will describe the improvements. The lines will each contain three integers, a, b ($1 \leq a < b \leq n$) and c ($1 \leq c \leq 1,000$), which indicates that the pipe connecting stations a and b has its capacity increased by c (if there is currently no pipe between a and b, then one is created with capacity c). Note that it is possible for an (a, b) pair to be repeated in this section.

Output

Output $k + 1$ integers, each on its own line, describing the maximum amount of water that can reach the mansion. The first number is the amount of water reaching the mansion in the initial configuration. The next k numbers are the amounts of water reaching the mansion after each improvement.

Examples

stdin	stdout
3 2 1 1 3 10 2 3 1 2 3 15	1 10
6 10 2 1 3 2 1 4 6 1 5 1 3 5 8 4 5 7 2 4 3 2 5 4 2 6 1 5 6 9 3 6 5 2 6 9 1 6 3	8 9 12

Problem L. Barbells

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second

Your local gym has b bars and p plates for barbells. In order to prepare a weight for lifting, you must choose a single bar, 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 balance; they must sum to the same weight. The combination of plates on either side might be different, but the total weight on either side must be the same. What weights are available for lifting?

Input

The first line of input contains two integers, b and p ($1 \leq b, p \leq 14$), representing the number of bars and plates. Then, there are b lines each containing a single integer x ($1 \leq x \leq 10^8$) which are the weights of the bars. After that, there are p lines each containing a single integer y ($1 \leq y \leq 10^8$) which are the weights of the plates.

Output

Output a sorted list of all possible lifting weights, one per line. There must be no duplicates.

Examples

stdin	stdout
2 5 100 110 5 5 1 4 6	100 110 112 120 122 130