Problem A: Awards

Time limit: 1 s
Memory limit: 512 MiB

A popular lottery game is performed by sequentially choosing 6 random numbers between 1 and 45 inclusive. An *award* number is then constructed by concatenating the last digits of the 6 numbers (in the order they were chosen). Prizes are awarded to those players whose lottery ticket serial number completely or partially matches the award number. The prize depends on the length of the common suffix between the award number and the serial number (also a 6 digit number) as in the example below.

Serial numbers	Prize
251239	1st prize
X51239	2nd prize
XX1239	3rd prize
XXX239	4th prize
XXXX39	5th prize
XXXXXX	nothing

Prize scheme for the serial number 251239

The left column contains the serial numbers with the letter `X` denoting arbitrary digits, while the right column denotes the name of the prize awarded to the matching serial numbers. Given the 6 numbers drawn, determine the prize for each of the 3 target serial numbers.

Input

The first line contains six different positive integers between 1 and 45 inclusive — the chosen random numbers. Each of the three following lines contains one integer with exactly 6 digits (possibly written with leading zeros) — the target serial number.

Output

For each of the three target serial numbers output a line containing the name of the prize awarded.

Example

input

```
12 35 1 2 23 39
151239
251229
251339
```

output

```
2nd prize
nothing
5th prize
```

input

```
5 45 35 25 15 1
555551
235551
555552
```

output

```
1st prize
3rd prize
nothing
```

Problem B: Border

Time limit: 1 s
Memory limit: 512 MiB

Luka started driving international routes with his truck. His biggest problem is the border with Slovenia. The border is a point of entrance into the Schengen Area, so every truck is thoroughly examined. Because of this, Luka always has to wait several hours there. To pass the time, he comes up with various logic and math games.

In one of them, Luka first reads the numbers off of n license plates and writes them down on a piece of paper. Then he tries to find an integer m greater than 1 such that all integers on the paper give the same remainder when divided by m. Luka tries to find as many such integers m as possible. Write a program that, given Luka's n integers, determines all such integers m.

Input

The first line contains the integer n ($2 \leq n \leq 100$) — the number of integers on paper. Each of the following n lines contains one integer between 1 and 10^9 inclusive. All these integers will be distinct. The input data will guarantee that at least one integer m will always exist.

Output

Output all integers m separated by spaces, in increasing order.

Example

input

```
3
6
34
38
```

output

```
2 4
```

input

```
5
5
17
23
14
83
```

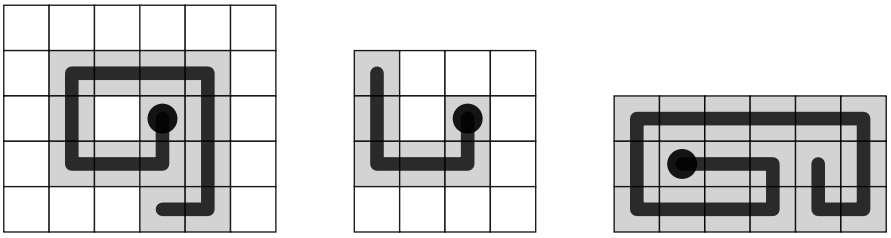
output

```
3
```

Problem C: Curl

Time limit: 1 s
Memory limit: 512 MiB

A snake is curled up and resting in a rectangular grid. The grid consists of n rows numbered 1 through n top to bottom and m columns numbered 1 through m left to right. The *snake body* is a sequence of different cells where each pair of subsequent cells in the sequence is adjacent in the grid (up, down,left or right). First cell in the sequence is called the *tail* while the last one is called the *head*.



Illustrations of the three examples below

Additionally, the snake is curled to the left — when we move through the sequence of cells from the tail to the head we always either go forward or make a left 90 degree turn. Additionally, the snake does not intersect itself — moving from the tail to the head we traverse each occupied grid cell once.

You are given a grid with the cells belonging to the head, the tail and the rest of the body clearly marked. Find one possible layout of the snake — the sequence of cells starting from the tail, ending at the head, traversing each body cell exactly once and only going straight or turning to the left. You may assume that a solution always exists.

Input

The first line contains the integers n and m ($1 \leq n, m \leq 100$) — the number of rows and colums. Each of the following n lines contains a string of length m — one row of the rectangular grid. Each string character is either a dot "`.`" representing an empty cell, a lowercase letter "`x`" representing a body cell, an uppercase letter "`R`" representing the tail or an uppercase letter "`G`" representing the head. There will be exactly one "`R`" and exactly one "`G`" in the grid.

Output

Output s lines where s is the length of the snake (the number of cells not denoted by a dot in the grid). The k-th line should contain two integers r_k and c_k — the row and the column number of the k-th cell of the snake body on the path from tail to head.

Example

input

```
5 6
......
.xxxx.
.x.Gx.
.xxxx.
...Rx.
```

output

```
5 4
5 5
4 5
3 5
2 5
2 4
2 3
2 2
3 2
4 2
4 3
4 4
3 4
```

input

```
4 4
R...
x.G.
xxx.
....
```

output

```
1 1
2 1
3 1
3 2
3 3
2 3
```

input

```
3 6
xxxxxx
xGxxRx
xxxxxx
```

output

```
2 5
3 5
3 6
2 6
1 6
1 5
1 4
1 3
1 2
1 1
2 1
3 1
3 2
3 3
3 4
2 4
2 3
2 2
```

Problem D: Drawing

Time limit: 1 s
Memory limit: 512 MiB

Mirko has created a digital artwork — a matrix consisting of characters "0" and "1". Now he wishes to find the largest rectangle whose edge consists only of "1" characters (other characters inside the rectangle can be arbitrary). Find the largest possible area of such a rectangle.

Input

The first line contains the integers n and m ($1 \leq n, m \leq 400$) – the number of rows and the number of columns in the matrix respectively. Each of the following n lines contains a string of length m — one row of the matrix.

Output

Output the area of the largest rectangle as described above.

Example

input

```
6 6
111000
111110
111010
010010
011110
000000
```

output

```
16
```

input

```
3 4
0000
0000
0000
```

output

```
0
```

input

```
3 3
000
001
000
```

output

```
1
```

Problem E: Elections

Time limit: 3 s
Memory limit: 512 MiB

It is election time! A total of v voters attend the election, each casting their vote for one of n political parties. A total of m officials will be elected into the parliament. The conversion from votes to parliament seats is done using the D'Hondt method with a 5% threshold. More precisely, suppose that the parties are numbered 1 through n and that they receive $v_1, v_2, \ldots, v_n$ votes respectively. Parliament seats are allocated as follows:

1. All parties that received strictly less than 5% of v votes are erased from the list of parties.
2. The parliament is initially empty i.e. every party has zero seats allocated.
3. For each party P, the quotient $q_P = v_P/(s_P + 1)$ is calculated, where v_P is the total number of votes received by party P, and s_P is the number of seats already allocated to party P.
4. The party with the largest quotient q_P is allocated one seat. If multiple parties have the same largest quotient, the lower numbered party wins the seat.
5. Repeat steps 3 and 4 until the parliament is full.

The votes are being counted and only part of the v votes has been tallied. It is known how many votes each party has received so far. Write a program that calculates for each party, among all possible outcomes of the election after all v votes are counted, the smallest number of seats the party can win.

Input

The first line contains the integers v, n and m ($1 \leq v \leq 10^7, 1 \leq n \leq 100, 1 \leq m \leq 200$) — the numbers of votes, parties and seats in the parliament. The second line contains n non-negative integers $v'_1, v'_2, \ldots, v'_n$ — v'_k is the number of votes received so far by party k. The sum of these numbers will be at most v.

Output

Output n integers on the same line — the smallest number of seats each party can win.

Example

input

```
20 4 5
4 3 6 1
```

output

```
1 0 1 0
```

input

```
100 3 5
30 20 10
```

output

```
1 1 0
```

Problem F: Fun

Time limit: 0.5 s
Memory limit: 512 MiB

Mirko and Slavko are playing a fun game. First, Mirko writes a sequence of n real numbers between -10.00 and 10.00, each with exactly two fractional digits, on a very long piece of paper. Slavko then crosses out some of the numbers and writes the remaining numbers on another piece of paper without any reordering. Now, Slavko is awarded a score that is calculated by multiplying every two neighboring numbers and adding the results of all multiplications. For example, if the final sequence is $3.00, 6.00, -1.00, 2.00$ then Slavko will be awarded 10 points. If there are less then 2 numbers in the final sequence, Slavko is awarded 0 points.

You are given the sequence of numbers Mirko has chosen. Find the maximal possible score Slavko can obtain.

Input

The first line contains the integer n ($3 \leq n \leq 30\,000$) – the length of the sequence. The k-th of the following n lines contains a floating point number x_k written with exactly two fractional digits ($-10.00 \leq x_k \leq 10.00$) — the k-th element of the sequence.

Output

Output the largest possible score as described above using *exactly four* fractional digits.

Example

input

```
3
8.52
1.00
9.61
```

output

```
81.8772
```

input

```
5
1.00
2.00
3.00
4.00
5.00
```

output

```
40.0000
```

input

```
3
-1.25
10.00
-1.00
```

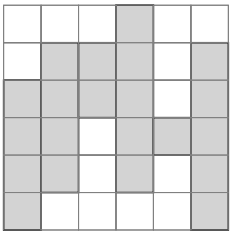
output

```
1.2500
```

Problem G: Grid

Time limit: 2 s
Memory limit: 512 MiB

Mirko and Slavko are playing with a game on a board shaped as a sequence of connected columns in a rectangular grid. Technically, a *board* is a sequence of n columns $c_1, c_2, \ldots, c_n$ where each column consists of one or more vertically connected cells. The column c_k is immediately to the right of the column c_{k-1} and the two always touch along at least one cell.



Initial board in the second example below

Assume the rows of the rectangular grid are denoted with integers starting from 1 and growing downwards, while the columns are denoted with integers starting from 1 and growing to the right. The column c_k is positioned in the k-th column of the rectangular grid and we define it by specifying l_k and h_k — the lowest and the highest row numbers among all of its cells.

Two types events can happen during a game:

1. The column t is changed, new l_t and h_t are specified.
2. We need to find the distance between two cells a and b on the current board. The distance is the minimal number of steps needed to move from a to b if, in each step, we can jump to the adjacent cell (up, down, left or right) inside the board.

As mentioned before, the columns c_k and c_{k-1} will always touch along at least one cell, and hence, there will always exists a path between any two cells inside the board.

Find answers to all the distance queries.

Input

The first line contains two integers n and q ($1 \leq n, q \leq 200\,000$) — the number of columns and the number of events. The k-th of the following n lines contains the integers l_k and h_k ($1 \leq l_k \leq h_k \leq 10^9$) — the row numbers of the topmost and the bottommost cell of the k-th column in the initial board.

Each of the following q lines contains several integers describing one event in one of the two possible formats:

- "1 t l_t h_t" — t the index of the column being changed and l_t and h_t are its new parameters ($1 \leq t \leq n, 1 \leq l_t \leq h_t \leq 10^9$).
- "2 c_a r_a c_b r_b" — the coordinates of the two cells a and b we would like to know the distance between ($1 \leq c_a, c_b \leq n, 1 \leq r_a, r_b \leq 10^9$). Both a and b will be inside the current board.

Output

Output p lines where p is the number of events of type 2 in the input. The k-th line should contain the distance between cells a and b in the k-th event of type 2.

Example

input

```
4 3
2 3
1 4
3 4
2 3
2 2 1 3 2
2 3 1 3 4
2 2 1 2 4
```

output

```
2
3
5
```

input

```
6 7
3 6
2 5
2 3
1 5
4 4
2 6
2 6 1 6 1
2 3 1 3 4
2 6 1 6 6
1 3 1 2
2 3 1 3 4
1 5 2 2
2 6 1 6 6
```

output

```
0
3
11
5
13
```

Problem H: Half

Time limit: 1 s
Memory limit: 512 MiB

Mirko and Slavko like to eat burek for breakfast. This time they bought one with cheese and another with meat. Watching the hot bureks on the table, teasing their nostrils with enticing aromas, they could not decide on who would get which burek so they decided to split both of them in half.

Mirko boasted that he could cut both bureks in two pieces of equal areas with just one swift stroke of his knife. Help Mirko accomplish this before the bureks get cold.

Each of the bureks is a convex polygon in the standard coordinate system. The bureks are positioned so that one of them is completely to left of the y-axis, and the other completely to the right. Given the coordinates of the polygons, find the equation of an arbitrary line that splits both bureks into two parts of equal areas.

You may assume that a solution always exists and that it is unique.

Input

The first line contains an integer n ($3 \leq n \leq 5\,000$) — the number of vertices in the first burek. Each of the following n lines contains two floating-point numbers x and y ($-1000 < x < 0, -1000 < y < 1000$) — the coordinates of one vertex in the burek. The following line contains an integer m ($3 \leq m \leq 5\,000$) — the number of vertices in the second burek. Each of the following m lines contains two floating-point numbers x and y ($0 < x < 1000, -1000 < y < 1000$) — the coordinates of one vertex in the burek.

In both bureks, the vertices will be given in the counter-clockwise order. The coordinates of the vertices will be given with exactly three digits after the decimal point. No three points in a polygon will be collinear.

Output

Output two floating-point numbers A and B so that $y = Ax + B$ describes a line Mirko can use to cut the bureks in half. The solution is considered correct if the absolute error of each printed value compared to the official solution is less than or equal to 0.001.

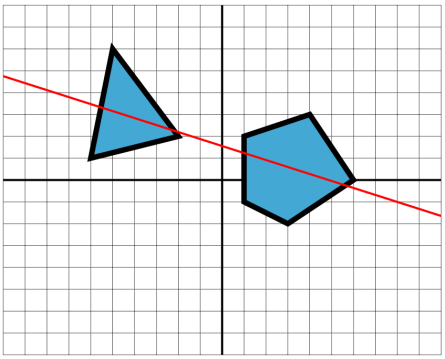
Example

input

```
3
-6.000 1.000
-2.000 2.000
-5.000 6.000
5
1.000 -1.000
3.000 -2.000
6.000 0.000
4.000 3.000
1.000 2.000
```

output

```
-0.319961 1.556489
```



input

```
4
-5.000 -1.000
-3.000 -1.000
-3.000 6.000
-5.000 6.000
4
3.222 2.000
5.000 1.000
5.000 4.000
3.222 3.000
```

output

```
0.000000 2.500000
```

Problem I: Indentation

Time limit: 1 s
Memory limit: 512 MiB

Python can be used at this year's Central Europe Regional Contest for the first time! Let us consider a simple imperative programming language that uses somewhat similar syntax. The language has only two commands:

- `P`x where x is a lowercase letter from the English alphabet: this command prints the letter x.
- `F`k where k is a positive integer less than or equal to 20 written without leading zeros: this command repeats the subsequent block of commands exactly k times.

Blocks of commands are defined by indentation levels — we use the dot characters "`.`" for indentation instead of the more usual spaces or tabs. Given a non-negative integer n, we define *n-level block* as a sequence of lines where each line is:

- `P` command indented with n dots, or
- `F` command indented with n dots followed by a m-indented block where m is greater than n.

You are given a *program* — a 0-indented block — find the number of appearances of each lowercase letter in its output.

Input

The first line contains an integer l ($1 \leq l \leq 100$) — the number of lines in the program. Each of the following l lines contains a string consisting of at most 80 characters — one line of the program. You may assume that the program is valid according to the rules above and that it prints at most 10 000 characters total.

Output

For each letter x that is printed by the given program, output a single line of the form "$x\ \ k_x$" where k_x is the number of appearances of letter x in the output. Lines should be ordered alphabetically by the letters x.

Example

input

```
5
Px
Py
F3
..Px
..Pz
```

output

```
x 4
y 1
z 3
```

input

```
9
F3
..Pa
..Pb
..F4
...Pc
..F2
.....Pd
.....Pe
..Pa
```

output

```
a 6
b 3
c 12
d 6
e 6
```

input

```
8
Pa
F3
.F12
..Px
..F20
...F2
....F3
.....Pa
```

output

```
a 4321
x 36
```

Problem J: Journal

Time limit: 1 s
Memory limit: 512 MiB

You are writing a tool that will help the *Journal of Chemistry* editors balance chemical equations in the papers they need to review. For the purpose of this problem we define the valid equations using a simple grammar below. In the grammar rules, [<item>] denotes an optional item while [<item>]* denotes an item that repeats zero or more times. Simply put, *factors* correspond to elements, *formulas* to molecules and *terms* to groups of molecules going in or out of a chemical reaction.

<equation> ::= <term> "=" <term>
<term> ::= <formula> ["+" <formula>]*
<formula> ::= <factor> [<factor>]*
<factor> ::= <uppercase> [<lowercase>]* [<nonzerodigit>]

Each factor is composed of a single uppercase letter, followed by zero or more lowercase letters and optionally a single digit "1"–"9". The letters in a factor represent the name of a chemical element and the digit the number of atoms of that element. If the digit is missing, the factor is composed of a single atom. For example, formula "NH3" represents a molecule with one atom of element "N" and three atoms of element "H".

Equation is *balanced* if the number of atoms of each element is the same on the left hand side and on the right hand side. If the equation is unbalanced, we can attempt to balance it by adding integer coefficients in front of formulas. If we place a coefficient a in front of a formula f that means that a molecules of f participate in the reaction — the number of atoms of all elements in f is multiplied by a. Coefficients can also be negative.

Find coefficients that balance a given formula. The absolute value of each coefficient can be at most 10^9 and at least one coefficient has to be different from zero. You may assume that a solution exists.

Input

The first line contains a string consisting of at most 100 characters — valid equation according to the rules above. No space characters appear inside an equation.

Output

Output n numbers on a single line where n is the total number of formulas in the given equation — the coefficients that need to be put in front of the formulas, left to right, in order to balance the equation.

Example

input	input	input
N2+H2=NH3	Al+O2=Al2O3	Na2S2O3+H2SO4=Na2SO4+H2O+SO2+S
output	**output**	**output**
1 3 2	4 3 2	1 1 1 1 1 1

Problem K: Kaktus

Time limit: 1 s
Memory limit: 512 MiB

Let us recall few standard definitions from graph theory. A *simple path* is a path which does not have repeating nodes. The *length* of the path is the number of nodes it contains. A *simple cycle* is a cycle which does not have repeating edges or nodes (other than the start and the end node). A *cactus graph* is a connected graph in which every edge belongs to at most one simple cycle.

You are given a cactus graph with n nodes and m edges. Find the number of simple paths of length l, for each l between 1 and n, modulo $10^9 + 7$.

Input

The first line contains two integers, n and m ($1 \leq n \leq 4\,000, 1 \leq m \leq 100\,000$). Each of following m lines contains two integers a and b ($1 \leq a < b \leq n$) which represent a bidirectional edge between nodes a and b. Every pair (a, b) appears at most once in the list of edges.

Output

Output n integers in a single line — the number of simple paths of length l, for each l between 1 and n.

Example

input

```
3 2
1 2
2 3
```

output

```
3 4 2
```

input

```
3 3
1 3
2 3
1 2
```

output

```
3 6 6
```

Problem L: Lanes

Time limit: 1 s
Memory limit: 512 MiB

The road network in a country consists of cities, represented by points in the standard coordinate system, and roads, represented by line segments that connect individual pairs of cities. Two roads may intersect, but in that case, overpasses and underpasses are built and, therefore, the road connecting cities A and B can be only used to travel from A to B and vice versa, even if the road intersects other roads or passes through other cities. The *length* of the road is the length of the corresponding line segment (i.e. the Euclidean distance between the two endpoint cities).

In the beginning, the road network consists of only two cities connected by road and grows as time passes: in each step one new city is added and is connected by two new roads with different two existing cities which are already directly connected by road.

Write a program that will simulate the growth of the road network and find answers to queries about the shortest path between two arbitrary cities. More specifically, your program must support the following commands:

- `d` x y A B – A new city is added at coordinates (x, y), and connected by two new roads with cities A and B (that are already directly connected with a road).
- `u` A B – Report the length of the shortest path between cities A and B.

The cities are denoted with integers starting from 1. The location of cities 1 and 2 is given and each new city being added is denoted with the subsequent integer. Cities 1 and 2 are also connected by road.

Input

The first line contains integers x_1, y_1 ($0 \leq x_1, y_1 \leq 10^6$) — the coordinates of city 1. The second line contains integers x_2, y_2 ($0 \leq x_2, y_2 \leq 10^6$) — the coordinates of city 2.

The third line contains the integer n ($1 \leq n \leq 10^5$) — the number of commands. Each of the following n lines contains one command. Each command is one of the following:

- "`d` x y A B" where x and y are integer coordinates of the city being added ($0 \leq x, y \leq 10^6$), and A and B different integers less than or equal to the current number of cities — the labels of two cities being connected by roads with the city being added. The cities A and B will always be already directly connected by road.
- "`u` A B" where A and B are different integers less than or equal to the current number of cities. This command means we are interested in the length of the shortest path between A and B.

No two cities will have the same coordinates.

Output

For each command of the type "`u`", you must output one integer – the distance between required cities. The answers to individual queries must be printed in the order of the queries in the input data.

The solution is considered correct if the absolute error of each printed value compared to the official solution is less than or equal to 0.1.

Example

input

```
6 4
10 4
9
d 6 7 2 1
u 1 2
u 3 2
d 10 2 1 2
u 3 4
d 12 7 2 4
u 5 3
u 4 5
u 1 5
```

output

```
4.000000
5.000000
7.000000
8.605551
5.385165
7.605551
```

input

```
1 1
3 8
10
d 8 2 1 2
u 2 1
d 2 9 1 3
u 1 4
d 4 7 3 4
u 4 3
d 6 1 5 4
u 6 5
d 0 0 4 6
u 7 3
```

output

```
7.280110
8.062258
9.219544
6.324555
18.439089
```