2023 Argentinian Programming Tournament (TAP)

A. Alfajores

1 second, 1024 megabytes

Seba is the manager of the Paper Airplanes Workshop (TAP, the acronym in Spanish), a very large company dedicated to the art of origami. TAP has a very large building with M offices. In the i-th office, there are E_i employees working.

Due to the high demand for their product, Seba travels constantly. When he returns from his trips, he usually brings a large box of alfajores to share with his employees.

To distribute them, he visits each of the M offices of the company in order, from 1 to M.

When he arrives at the i-th office, he distributes as many alfajores as possible equally among the E_i employees of the office. After distributing them, he takes the box with the remaining alfajores and moves on to the next office.

Once he has visited all the offices, he sits at his desk and enjoys the remaining alfajores.

Seba is afraid of overindulging in sweets, so he needs to know how many alfajores he has consumed. The problem is that he does not keep track of the amount left in the box after each distribution corresponding to each trip. Luckily, he has the N tickets corresponding to the alfajores purchases, and since he knows how many people work in each office, he is sure that you can calculate those quantities for him.

Input

The first line contains two integers N and M ($1 \leq N, M \leq 10^5$), representing the number of trips Seba made and the number of offices in TAP.

The second line contains N integers $A_1, A_2, \ldots, A_N$ ($1 \leq A_i \leq 10^9$), where A_i is the number of alfajores Seba bought on the i-th trip.

Finally, the third line contains M integers $E_1, E_2, \ldots, E_M$ ($1 \leq E_i \leq 10^9$), where E_i is the number of people working in the i-th office.

Output

A single line with N integers, the quantities of alfajores that remained in the box after the distribution corresponding to each of Seba's trips.

input
3 3 140 79 5 90 42 5
output
3 2 0

input
10 8 120 456 7458 84 123 84 213 185 987 654 97 73 61 41 52 23 11 7
output
0 0 2 0 3 0 1 4 6 0

In the first example, Seba brings 79 alfajores on the second trip and when passing through the first floor, as there is not enough for each employee to take one, none of them receive alfajores. When passing through the second floor, each one takes 1 alfajor, leaving 37 in the box. Finally, he arrives at the third floor where each employee takes 7, leaving 2 alfajores for Seba.

B. Black and white

1 second, 1024 megabytes

Fernanda is playing with tiles on a board. The board is an $N \times N$ grid, with some squares painted white and others black. All squares have size 1cm $\times$ 1cm.

Fernanda will use rectangular domino tiles, of size 1cm $\times$ 2cm. She will place them in a **horizontal** position, occupying **exactly two horizontally neighboring squares** of the board.

The game imposes the rule that she can only place a domino tile on two black squares. White squares are strictly forbidden. Additionally, the same square cannot be occupied by more than one tile.

Given the colors of the $N \times N$ squares, can you help Fernanda by indicating the maximum number of tiles she can place on the board?

Input

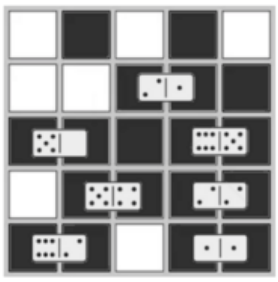
The first line contains an integer N ($1 \leq N \leq 50$), the size of the board. N lines follow describing the board. The i-th line contains a string of N characters N or B, indicating the colors of the squares in the i-th row, from left to right. N for black ("Negro" in Spanish) and B for white ("Blanco" in Spanish).

Output

A single line with an integer indicating the maximum number of tiles that can be placed on the board.

input
5 BNBNB BBNNN NNNNN BNNNN NNBNN
output
7

In the example, it is possible to place 7 tiles on the board as shown in the following figure. There is no way to place 8 or more tiles, so the answer is 7.



C. Chromatic

2 seconds, 1024 megabytes

A group of N friends is going to paint their houses violet. They have N cans of blue paint and N cans of red paint. They are going to distribute the paint so that each person receives one can of each color. It doesn't matter how much paint of each color they receive, but they want the distribution of the total paint to be as equitable as possible. That is, they want the difference between the person who receives the most paint and the one who receives the least to be as small as possible.

Given the amount of paint in each can, what is the minimum difference?

Input

The first line contains an integer N ($2 \leq N \leq 10^5$), the number of friends.

The second line contains N integers: $A_1, \ldots, A_N$ ($1 \leq A_i \leq 10^9$), the amount of paint in each of the N cans of red paint.

Finally, the third line contains N integers: $B_1, \ldots, B_N$ ($1 \leq B_i \leq 10^9$), the amount of paint in each of the N cans of blue paint.

Output

A single line with an integer indicating the smallest possible difference between the amount of paint received by the person who receives the most paint, and the amount received by the person who receives the least.

input
2 1 2 1 2
output
0

input
3 1000000000 1 1000000000 1000000000 1 1000000000
output
999999999

In the first example, the optimal paint distribution can be achieved as follows: The first friend receives 1 unit of blue paint and 2 units of red paint, which is a total of 3 units of paint. The second friend receives 1 unit of red paint and 2 units of blue paint, which is also a total of 3 units of paint. Therefore, the difference between the person who receives the most paint and the one who receives the least paint is $3 - 3 = 0$.

D. Assigning problems

0.75 seconds, 1024 megabytes

As a new initiative, the AAPC (Argentine Competitive Programming Association, by its initials in Spanish) aims to organize many contests in a year. Organizing a *contest* consists of selecting a group of *problems* of K different *levels* numbered from 1 to K in increasing order of difficulty. Moreover, every contest must have exactly C_i problems assigned to level i for each i between 1 and K.

In addition to that, each problem has a certain *grade of difficulty*. If a problem has a grade of difficulty d, it means that the AAPC can assign that problem in a contest in any level greater than or equal to d.

Naturally, the AAPC is not able to repeat problems among different contests, that means that a single problem can only be assigned to a single contest. Moreover, a single problem cannot be assigned to more than one level within the same contest.

The AAPC has requested problem proposals among its members. In total the AAPC has collected P_i problem proposals with a grade of difficulty equal to i for each i between 1 and K, and plans to use them to organize as many contests as possible.

Using these problems (not necessarily all of them), what is the maximum number of contests that the AAPC is able to organize?

Input

The first line contains an integer $K (1 \leq K \leq 10^5)$, which represents the number of different levels in the contests organized by the AAPC.

Then, a second line with K integers $C_1, C_2, \ldots, C_K$, where each $C_i (1 \leq C_i \leq 10^9)$ indicates the number of problems assigned to level i that a contest organized by the AAPC must have.

Finally, a third line with K integers $P_1, P_2, \ldots, P_K$, where $P_i (0 \leq P_i \leq 10^9)$ denotes the number of problem proposals with a grade of difficulty of i that the AAPC has at their disposal to create the desired contests.

Output

A single line with an integer that represents the maximum number of contests that the AAPC can organize while attaining the required amount of problems per level using the problems that they have at their disposal.

input
5 2 2 2 2 2 100 0 0 0 0
output
10

input
5 2 3 2 2 2 10 0 9 3 1
output
2

input
5 2 3 2 2 2 0 100 7 0 6
output
0

In the first example, each contest consists of 5 different levels, and 2 problems need to be assigned to each level. The AAPC has 100 problems at their disposal, each one of them with a grade of difficulty of 1, that can be assigned to any level. Hence, the association can organize a maximum of 10 contests using those problems.

E. Finding progressions

1 second, 1024 megabytes

Catalina needs help with her Math homework, which consists of finding all the *increasing arithmetic progressions of integers* (*) that meet the following conditions:

- The progression contains the number A
- The sum of the terms of the progression is S
- All terms of the progression are in the range $[L, R]$

Catalina is a responsible girl and doesn't want someone else to solve the task for her, but she could use a little help. What Catalina needs is for you to tell her how many progressions exist under these conditions, so that she could know when to stop searching.

(*) An *increasing arithmetic progression of integers* is a non-empty list of n integer numbers $x_1, x_2, \ldots, x_n$ where $x_i - x_{i-1} = D > 0$ for every integer i such that $2 \leq i \leq n$, where D is a positive integer. For example, $[7]$, $[1, 2, 3]$ and $[2, 4, 6, 8]$ are valid progressions, but $[5, 1]$, $[2, 4, 8]$ and $[3, 6, 9, 6]$ are not.

Input

A single line with four integers A, S, L and R ($1 \leq L \leq A \leq R \leq 10^{12}$, $1 \leq S \leq 10^{18}$, $0 \leq R - L \leq 10^5$).

Output

A single line with an integer indicating how many progressions meet the conditions described by the statement.

input
5 15 1 10
output
6

input
5 15 2 10
output
4

input
7 30 3 26
output
4

input
5 5 5 5
output
1

The 6 valid progressions for the first example are: $[5, 10]$, $[1, 5, 9]$, $[2, 5, 8]$, $[3, 5, 7]$, $[4, 5, 6]$ and $[1, 2, 3, 4, 5]$

F. Cold day at the beach

0.5 seconds, 1024 megabytes

A group of friends, already retired from competitive programming, spontaneously decided to take a week-long vacation in the coastal city of Mar de AJI. Since they didn't check the weather forecast, they found cloudy and windy days when they arrived. However, the weather conditions were not going to ruin their vacation.

To have fun at the beach, they decided to organize a *tejo* tournament between two teams, named A (blue team) and R (red team). Tejo is played on a rectangular court marked out on the sand, which is W centimeters wide and L centimeters long. The corners of this court have coordinates $(0, 0)$, $(W, 0)$, $(0, L)$, and (W, L).

Initially, there is a *tejín* at position (T_x, T_y). Then, each team takes turns making N throws of discs (called *tejos*) trying to get as close to the tejín as possible, even landing in the same position as the tejín, located above it.

At the end of the N throws, the team whose tejo is closest to the tejín is the winning team. Furthermore, the winning team receives one point for each tejo that is closer to the tejín than the closest tejo from the opposing team. The distance of a tejo to the tejín is measured as the euclidean distance from the center of the tejín to the center of such tejo.

In addition to the team's throws, the position of the center of the tejín is also known. It is guaranteed that all throws are within the court boundaries (or on the line) and that **there are no two tejos at the same distance from the tejín** in the given layout. You are asked to calculate which team won and also how many points they earned.

Input

The first line contains an integer $N (1 \leq N \leq 1000)$, that represents the number of throws made by each team.

Then, the second line contains four integers W, L, T_x, and T_y. The first two correspond to the width and length dimensions of the court in centimeters $(1 \leq W \leq 10^4, 1 \leq L \leq 10^4)$, and the last two indicate the location of the tejín $(0 \leq T_x \leq W, 0 \leq T_y \leq L)$.

After that, N lines follow, each with two integers describing a throw by team A. The i-th line contains two integers X_i, Y_i where X_i indicates the width location of the i-th throw $(0 \leq X_i \leq W)$, and Y_i denotes the length location of the i-th throw $(0 \leq Y_i \leq L)$ by team A.

Finally, there are another N lines that describe the throws of team R in the same way.

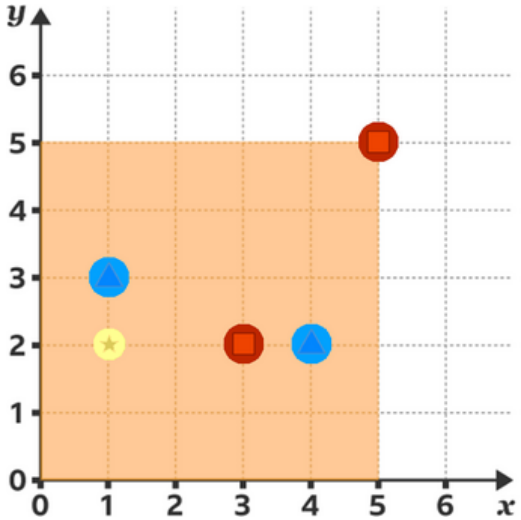
Output

A single line with two values separated by a space. The first value must be A or R depending on which team emerged as the winner, and the second value represents the number of points received by that team at the end of all throws.

input
2 5 5 1 2 1 3 4 2 3 2 5 5
output
A 1

input
3 10 10 0 5 0 0 0 2 0 4 0 1 0 3 0 5
output
R 1

input
3 10 10 0 5 0 3 0 4 0 5 0 0 0 1 0 2
output
A 3



The image corresponds to the first example. Each team made two throws. The blue tejos (with a triangle inside) belong to team A in positions $(1, 3)$ and $(4, 2)$. The red tejos (with a square inside) belong to team R in positions $(3, 2)$ and $(5, 5)$. The court is 5 centimeters wide and long. The tejín (yellow disc, with a star inside) is located at position $(1, 2)$. The distances from the throws to the tejín are $1, 3, 2$, and 5, where the first two throws correspond to team A and the last two to team R. Team A emerges as the winner and receives only 1 point (corresponding to the tejo that is 1 centimeter away from the tejín).

G. Great Heights

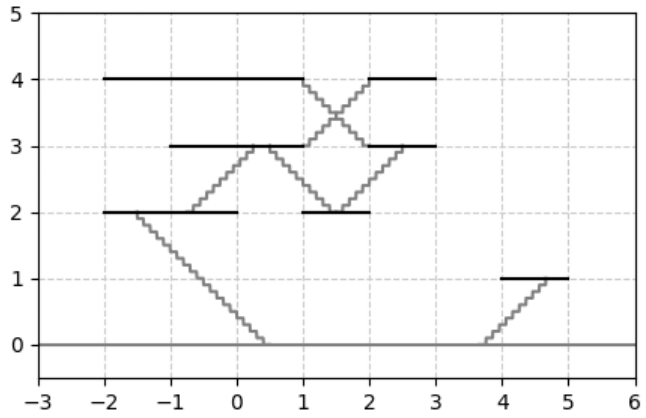
1.5 s, 1024 megabytes

In Nlogonia, the construction of the long-awaited Tall And Prestigious tower (TAP) has finally begun. The scaffolding that will be used by the workers in the various operations required to carry out this monumental work is already installed and ready for use. Each scaffolding can be modeled as a horizontal segment (i.e., aligned with the x-axis). The i-th scaffolding is located at a height H_i above the ground and, and covers the coordinates on the x axis between L_i and R_i inclusive.

However, the placement of the stairs that will serve for the access of the workers and transportation of materials has not yet been determined. For this reason, we have been hired to carry out this important task. Each staircase can be modeled as a segment with an upper and lower end. We call the lower end of a staircase its base. The bases of the stairs can be supported on the ground or on a scaffolding, and their upper ends must be supported on another scaffolding.

Furthermore, in order for them to be functional, the stairs must be constructed in such a way that it is possible to access from the ground (modeled by the x-axis, at height 0) to any of the scaffolding, going up and down as many stairs as necessary. Since the stairs are intended to be used for carrying heavy objects, it was decided that they should be installed at a 45° angle, so that if a staircase has the upper end at a height D meters above its base, its upper end will have an x coordinate that is D meters to the left or right of the x coordinate of its base.

It is possible to install the stairs in such a way that their segments intersect, but for safety reasons it is strictly forbidden to move from one staircase to another outside of a scaffolding. In other words, a staircase only connects the scaffolding on which its ends are supported (or the ground with a scaffolding, in the case of stairs with a base on the ground).



Since the company wants to save costs, they have asked us to determine the minimum total cost of installing all the necessary stairs, taking into account that the cost of a staircase is equal to the difference between the height of its upper end and the height of its base.

Input

The first line contains a single integer N $(1 \leq N \leq 10^5)$, the number of scaffolding installed in the TPT. The following N lines contain 3 integers H_i, L_i, and R_i $(1 \leq H_i \leq 10^9, -10^9 \leq L_i < R_i \leq 10^9)$, where H_i is the height of the i-th scaffolding, while L_i and R_i are the positions of the left and right ends of the scaffolding, respectively. It is guaranteed that the scaffolding do not share any points.

Output

A single line with an integer equal to the minimum total cost of installing all the stairs.

input

```
7
2 -2 0
3 -1 1
3 2 3
4 -2 1
4 2 3
2 1 2
1 4 5
```

output

```
8
```

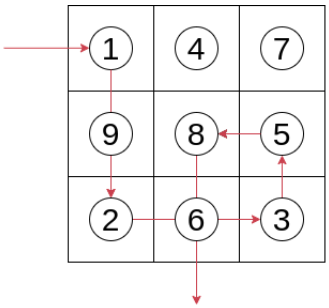
One way to obtain the result of the example can be found in the image in the problem statement, using 6 stairs with a cost of 1, and one staircase with a cost of 2. Note that in this solution, the stairs with a base on scaffolding located at a height of 3 are not connected to each other.

H. Robotic Skills

2 s, 1024 megabytes

Franco is learning robotics and he is very excited because he has already built his first robot. His original plan was for this robot to be able to play Hopscotch, but since this is very difficult, he settles for a simpler version of this game.

On the floor, there is a board with $N \times N$ squares drawn. Each of these squares contains an integer between 1 and N^2. It is also known that there are no repeated numbers. The robot can start its journey at any point outside the board and can only move horizontally and vertically, that is, in directions parallel to the edges of the board. In order to change its direction of movement from vertical to horizontal (or vice versa), the robot must be inside one of the squares. The robot can change its direction at most once in each square. It can never change its direction by 180°. The journey must end outside the board. Let $c_1, c_2, c_3, \ldots, c_k$ be the numbers written in the squares where the robot changed its direction. For the journey to be valid, it must satisfy $c_1 < c_2 < c_3 < \ldots < c_k$. The score of a journey is the number of squares in which the robot changed its direction. A valid journey on a 3×3 board is shown below:



Help Franco program the robot so that the score of his journey is as high as possible.

Input

The first line contains an integer N $(2 \leq N \leq 1000)$.

Then, the i-th of the following N lines contains N integers $A_{i,1}, A_{i,2}, \ldots, A_{i,N}$, which are the numbers written in the i-th row of the board $(1 \leq A_{i,j} \leq N^2)$. It is guaranteed that all numbers are different.

Output

A single line with the maximum score that the robot can achieve in its journey.

input

```
2
1 2
3 4
```

output

```
3
```

input

```
3
1 4 7
9 8 5
2 6 3
```

output

```
5
```

In the second example, a possible journey of the robot that achieves the maximum score can be found in the image in the problem statement. Note that the robot does not change direction in the squares with the numbers 6 and 9.

Another possible journey that also achieves the maximum score is obtained by changing direction in the squares $1, 2, 3, 5$ and 9.

I. Regional Integration

7.5 seconds, 1024 megabytes

The distant and modern city of Extremopolis is located right on the triple border between the nations of Circland, Squareland, and Triangland.

For this reason, the city has many buildings. These buildings can only be of 3 possible types:

1. The base of the building is circle-shaped.
2. The base of the building is square-shaped (**not necessarily aligned with the axes**). A square is a quadrilateral with all 4 sides equal and all 4 angles right angles.
3. The base of the building is triangle-shaped (**not necessarily aligned with the axes**).

You have a detailed map of the city, which shows the circles, squares, and triangles representing the bases of the different buildings. Two of these figures never intersect or touch each other, not even at a vertex, as they are different buildings.

The Institute of Coordination and Planning of Circland (ICPC) plans to open two new business offices: one should be located in a building with a triangular base, and the other in a building with a square base. With this, the institute aims to improve its commercial relations with neighboring countries.

Another extreme characteristic of Extremopolis is that it is located on the equator, has a very dry climate, and practically no cloudy days. Therefore, the level of ultraviolet radiation from the sun that its inhabitants experience is extremely high. Taking this into account, the ICPC wants to select the two buildings where it will open the new offices in such a way that walking from one to the other requires walking in the sun as little as possible.

Extremopolis has no streets or obstacles that obstruct the way, so it is possible to walk freely through any point in the city. All buildings are designed with a spacious and obstacle-free ground floor, so it is even possible to walk freely through the base of the buildings (i.e., the entire interior of the circle, square, or triangle), and these walking distances inside the buildings do not count towards the total distance walked under the sun.

Your task is to compute the minimum distance that needs to be walked under the sun to travel between both offices, if they are opened in the ideal buildings to minimize that distance.

Input

The first line contains three integers C, Q, T: the number of buildings with circular base ($0 \le C$), square base ($1 \le Q \le 10^5$), and triangular base ($1 \le T \le 10^5$) respectively.

It is also known that $(T + C)(Q + C) \le 10^6$.

Each of the following C lines describes a building with a circular base using 3 integers x, y, r. The corresponding circle has its center at (x, y) and radius r.

Each of the following Q lines describes a building with a square base using 4 integers x_1, y_1, x_2, y_2. The corresponding square has a pair of **opposite** vertices at (x_1, y_1) and (x_2, y_2). It is guaranteed that these are two distinct points in the plane, i.e., $(x_1, y_1) \neq (x_2, y_2)$.

Each of the following T lines describes a building with a triangular base using 6 integers $x_1, y_1, x_2, y_2, x_3, y_3$. The corresponding triangle has its three vertices at the points (x_1, y_1), (x_2, y_2), and (x_3, y_3). It is guaranteed that these are three distinct points in the plane, and they are not collinear (the triangle is not degenerate).

The radii r and all coordinates x, y are integers between 1 and 10^9 inclusive.

It is guaranteed that there will be no two buildings whose bases intersect, not even at a point on their boundary.

Output

A single line with a single number: the minimum distance that needs to be walked under the sun while fulfilling the requirements.

This answer will be accepted if it has a relative or absolute error less than or equal to 10^{-6}.

Formally, let a be your answer and b be the jury's answer. Then, your answer will be accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

input

```
2 1 1
10 1 5
20 1 4
1 1 4 4
24 5 28 50 45 4
```

output

```
3.656854249
```

input

```
0 1 1
1 1 4 4
5 1 5 10 10 1
```

output

```
1.0
```

input

```
0 2 2
1 1 10 2
3 12 2 20
50 10 50 100 100 10
450 410 450 4100 4100 410
```

output

```
40.792156108
```

In the first two examples, there is only one building with a square base and one building with a triangular base, so those must be chosen to open the offices.

In the first example, to walk the shortest distance under the sun, it is convenient to pass through the buildings with circular bases.

Note that in the third example, the squares are not aligned with the axes.

J. Jester in danger

0.75 seconds, 1024 megabytes

In the kingdom of Graphland, there are N cities, identified by consecutive integers from 1 to N, inclusive. Graphland is a very peculiar kingdom because it has two capitals: City 1 is the royal capital of Graphland, and City N is the constitutional capital.

In Graphland, there are also M bidirectional roads. Each road directly connects exactly two different cities. The i-th road connects cities A_i and B_i.

Let's call G_0 to the map of Graphland, with its N cities and M roads. In order to prepare contingency plans and increase the security of the kingdom, the king is interested in studying what would happen if the kingdom were to lose cities successively as a result of possible catastrophes. In the hypothetical scenario that the king wants to study, K cities will be eliminated in order, one after another: $C_1, C_2, \ldots, C_K$. Starting with the map G_0, after each city is eliminated, the new maps $G_1, G_2, \ldots, G_K$ are obtained, respectively. In other words, G_i is obtained by removing city C_i (along with any road that connects to that city) from map G_{i-1}, for each $1 \leq i \leq K$. None of the eliminated cities will be a capital.

For each $0 \leq i \leq K$, a capital path in G_i is a sequence of cities in map G_i that starts at city 1, ends at city N, and such that consecutive cities in the sequence are directly connected by a road. The length of a capital path is the number of cities in the sequence. If there exists a capital path in G_i, the king defines $D(i)$ as the minimum length among all capital paths in G_i. It is known that there exists a capital path in G_0, so $D(0)$ is defined.

For each $0 \leq i \leq K$, the king says that G_i is broken if there is no capital path in G_i, or if there is one but $D(i) > D(0)$, meaning that the minimum length of a capital path in map G_i is strictly greater than in G_0.

For each $0 \leq i \leq K$ such that G_i is not broken, the king says that a city v ($2 \leq v \leq N-1$) in map G_i is critical if the map obtained by removing v from G_i would be broken.

You are a jester in the king's court, and unfortunately, you are also the only person in the entire kingdom who knows how to code. To avoid the punishment of the relentless king of Graphland, you must write a program that, for each $0 \leq i \leq K$, determines the number of critical cities in map G_i, or indicates that G_i is broken.

Input

The first line contains three integers N, M, K ($3 \leq N \leq 10^5$, $1 \leq M \leq 2 \cdot 10^5$, $1 \leq K \leq N-2$).

Then M lines describe the roads. The i-th line contains two integers A_i and B_i ($1 \leq A_i < B_i \leq N$). These M lines are all distinct.

Then K lines contain the cities to be eliminated, in the order they are eliminated. That is, the i-th line contains the integer C_i ($2 \leq C_i \leq N-1$). These K cities are all distinct.

Output

A single line with $K+1$ integers, indicating what the king requested for each map.

More precisely, for each $0 \leq i \leq K$, the i-th integer (counting from 0) should be:

- -1, if map G_i is broken
- The number of critical cities in map G_i, if map G_i is not broken

input

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

output

```
0 1 -1
```

input

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

output

```
2 2 2
```

input

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

output

```
1 -1 -1 -1
```

1. In the first example, the capital paths of minimum length in G_0 are $1 \to 2 \to 4$ and $1 \to 3 \to 4$. If a non-capital city, either 2 or 3, were to be eliminated, there would still be at least one capital path with the same minimum length, so there are no critical cities in G_0. In G_1, city 3 has already been eliminated, so if city 2 were to be eliminated in G_1, there would be no remaining path between capitals and the map would be broken. For this reason, city 3 is critical in G_1, and there is exactly one critical city in G_1. Finally, in G_2, both cities (2 and 3) have been eliminated and there are no remaining capital paths, so G_2 is broken.
2. In the second example, 2 and 3 are the only critical cities in G_0, G_1, and G_2.
3. In the third example, 2 is the only critical city in G_0, and G_1, G_2, G_3 are broken. Note that in G_1 there exists a capital path $1 \to 3 \to 4 \to 5$, but its length (and the length of any other) is greater than the capital path of minimum length in G_0, which is $1 \to 2 \to 5$.

K. Kitties

7 seconds, 1024 megabytes

Marcos is a veterinarian and cats are his favorite animal. So much so, that for a few years now he has had a pet shop called Kitties, where he sells all kinds of things for cats. In his shop, there is also a screen of height H_1 and width W that displays a website that Marcos created, with different kitten pictures that get renewed from time to time.

The website consists of a vertically scrollable panel, which has the same width W as the screen, but has a greater height $H_2 > H_1$ (that is why it is vertically scrollable).

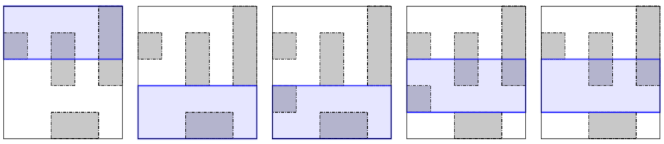
Initially, the panel contains N kitten pictures placed in different positions. K new pictures will appear later, so there are $N+K$ possible different pictures in total.

The i-th of these pictures is identified with the positive integer i, called its id. Pictures with id from 1 to N are the initial ones, and pictures with id $N+1$ to $N+K$ will be added later. For each i from 1 to $N+K$, the picture with id i covers a rectangular portion of the panel defined by the following values:

- U_i: the distance from the **top** edge of the panel to the **top** edge of the picture
- D_i: the distance from the **top** edge of the panel to the **bottom** edge of the picture
- L_i: the distance from the **left** edge of the panel to the **left** edge of the picture
- R_i: the distance from the **left** edge of the panel to the **right** edge of the picture

As the panel is bigger than the screen, then only a subrectangle of it will be visible at all times. Similar to the pictures, such subrectangle is defined by the values s_U, s_D, s_L and s_R.

Initially, the *scroll position* of the website is $x = 0$, meaning that the subrectangle of the panel visible in the screen is $s_U = 0$, $s_D = H_1$, $s_L = 0$, $s_R = W$. However, the website is programmed so that from time to time, some new pictures appear, others disappear, and the scroll position changes.



Marcos is afraid that these changes programmed to happen automatically could affect the efficiency of the website, as the pictures are loaded and removed from memory depending on whether they are visible or not. And that's why he needs your help.

Given Q changes that happen on the website in order, for each of them you have to answer the following two questions:

- How many pictures that were **not visible** (partially or completely) before this change, will be **visible** after the change?
- How many pictures that were **visible** (partially or completely) before this change, will **not be visible** after the change?

Input

A first line with five integers N, Q, H_1, H_2 and W ($1 \le N, Q \le 10^5$, $1 \le H_1, H_2, W \le 10^9$, $H_1 < H_2$) explained in the statement.

Then N lines follow. The i-th of these lines contains four integers U_i, D_i, L_i and R_i, describing the location in the panel of the picture with id i.

Then Q lines follow that represent the changes that happen on the website in order. Each of these Q lines has a character followed by a number of integers, according to one of the following formats:

- A j U_j D_j L_j R_j: indicating that a new picture appears with id j, located in the panel as indicated by U_j, D_j, L_j and R_j. The i-th change of this type in the input will have $j = N + i$, and there will be K changes of this type in total.
- D j $(1 \le j \le N + K)$: indicating that the picture with id j disappears (it is guaranteed that this picture is present in the panel).
- M x $(0 \le x \le H_2 - H_1)$: indicating that the scroll position is modified to take the value x, meaning that the subrectangle of the panel that is displayed in the screen changes to be $s_U = x$, $s_D = x + H_1$, $s_L = 0$, $s_R = W$

It is guaranteed that the pictures don't overlap nor touch each other, and for all i from 1 to $N + K$, $0 \le U_i < D_i \le H_2$, $0 \le L_i < R_i \le W$.

Output

Q lines, one for each change given in the input, with two integers answering the two questions present in the statement, in the order in which these questions were asked.

input
4 4 2 5 5 1 2 0 1 1 3 2 3 0 3 4 5 4 5 2 4 M 3 A 5 3 4 0 1 M 2 D 5
output
1 3 1 0 2 1 0 1

The five images in the statement reflect the initial configuration of the example and the configuration after each of the changes, in order. Note that the blue rectangle in each image represents the subrectangle of the panel visible on the screen, while the gray rectangles correspond to the pictures.

The ids of the pictures visible in each of these five moments are: $[1, 2, 3]$, $[4]$, $[4, 5]$, $[2, 3, 5]$ and $[2, 3]$

L. Game series

1 second, 1024 megabytes

Two teams, "Archimedians F.C." and "Pithagoreans F.C.", compete in the final of a football tournament, playing a **two-game series**. Each game ends with a certain number of goals scored by each team.

After these two games, a team is the winner of the series if they scored **more goals in total** than their opponent. Otherwise, if they finish with the same number of goals at the end of both games, the series must be decided in a third tiebreaker game.

Knowing the results of each of these two games (i.e., how many goals each team scored in each game), you are asked to write a program that determines which team won the series or if the series should go to a tiebreaker game.

Input

A first line with two integers $A_1, P_1 (0 \le A_1, P_1 \le 31)$, which represent the number of goals scored by "Archimedians F.C." and "Pithagoreans F.C." in the first game, respectively.

Then, a second line, similar to the first, with two integers $A_2, P_2 (0 \le A_2, P_2 \le 31)$, indicating the number of goals scored by "Archimedians F.C." and "Pithagoreans F.C." in the second game, respectively.

Output

A single line with a single character. In case the team "Archimedians F.C." wins the series, you should write A. If the team that wins the series is "Pithagoreans F.C.", you should print P, and if a tiebreaker game is needed to determine the series, you should print the character D.

input
3 1 1 1
output
A

input
4 3 1 3
output
P

input
2 4 2 0
output
D

In the first example, "Archimedians F.C." scored 3 goals in the first game and 1 goal in the second game, totaling 4 goals in the series. On the other hand, "Pithagoreans F.C." scored 1 goal in both games, totaling 2 goals in the series. Therefore, "Archimedians F.C." emerged as the winning team of the series, and the answer is **A**.

M. Multiple Downloads

0.5 seconds, 1024 megabytes

CDownloader is a download manager written in the C language. In other words, it is a program capable of automatically downloading a list of N files from the internet, all at once.

Each file to be downloaded can be either prioritized or not. Unlike other much more complex managers with fairer and better rules, CDownloader uses a very simple rule: at any given moment, from the total download speed of T megabytes per second (MB/s) that the user's connection has, it will use 75% of that speed to download the prioritized files, and 25% to download the non-prioritized files. Within each group, the assigned download speed is distributed equally.

For example, if CDownloader needs to download the following files:

- A, prioritized, with size 10 MB
- B, non-prioritized, with size 100 MB
- C, prioritized, with size 20 MB
- D, prioritized, with size 200 MB
- E, non-prioritized, with size 300 MB

And the user's connection speed is 40 MB/s, then initially CDownloader:

- Downloads A, with speed 10 MB/s
- Downloads B, with speed 5 MB/s
- Downloads C, with speed 10 MB/s
- Downloads D, with speed 10 MB/s
- Downloads E, with speed 5 MB/s

Since 75% of 40 MB/s is 30 MB/s and that is used for the 3 prioritized files, each one is downloaded at 10 MB/s. Also, 25% of 40 MB/s is 10 MB/s, and that is used in total for the 2 non-prioritized files, resulting in 5 MB/s for each one.

After 1 second of downloading, the download of file A is completed, and the new situation of the remaining files is:

- Downloads B (already 5 MB downloaded), with speed 5 MB/s
- Downloads C (already 10 MB downloaded), with speed 15 MB/s
- Downloads D (already 10 MB downloaded), with speed 15 MB/s
- Downloads E (already 5 MB downloaded), with speed 5 MB/s

And so on until all downloads are completed. If at any time there are only prioritized files, or only non-prioritized files, then the download speed is distributed equally among all remaining files.

Your task is to write a program that calculates the total time CDownloader will take to download all the files from the internet.

Input
The first line contains two integers N and T ($1 \leq N \leq 50$, $1 \leq T \leq 1024$), the number of files to download and the download speed of the user's connection in MB/s.

Then N lines, each describing one of the N files. The i-th line contains an uppercase letter C_i ($C_i \in \{\texttt{P}, \texttt{N}\}$) and an integer X_i ($1 \leq X_i \leq 512$), indicating whether the i-th file is prioritized (letter P) or non-prioritized (letter N), and the size in MB.

Output
A single line with a single number indicating the total time in seconds until CDownloader completes the download of the N files.

This answer will be accepted if it has a relative or absolute error less than or equal to 10^{-4}.

Formally, let a be your answer and b be the jury's answer. Then, your answer will be accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \leq 10^{-4}$.

input
```
5 40
P 10
N 100
P 20
P 200
N 300
```

output
```
15.75
```

input
```
1 1
N 64
```

output
```
64
```

input
```
1 113
P 355
```

output
```
3.1415929204
```

N. Lucky Number

0.5 seconds, 1024 megabytes

Matías has a deck of N cards. Each card has an integer written on it. As his lucky number is five, he wants to form several groups with these cards in such a way that the sum of the numbers on the cards in each group is a multiple of five. No card can be in two or more groups. It is possible that some cards do not belong to any group. What is the maximum number of groups that Matías can form?

Input
A line with an integer N ($1 \leq N \leq 2 \cdot 10^5$), the number of cards in Matías' deck.

Then, a second line with N integers $C_1, C_2, \ldots, C_N$. Each one indicates the number written on the i-th card ($1 \leq C_i \leq 10^9$).

Output
A single line with a single integer, representing the maximum number of groups of cards that Matías can form.

input
```
6
33 21 66 8 1 108
```

output
```
1
```

input
```
9
1 6 7 10 16 18 41 42 77
```

output
```
3
```

input
```
2
19 7
```

output
```
0
```

In the first example, Matías can form a group with the following four cards: $33, 21, 8$, and 108. Their sum is $33 + 21 + 8 + 108 = 170$ which is a multiple of five. Two groups of cards whose sum is a multiple of five cannot be formed.

In the second example, Matías can form three groups as follows:

- The first group contains the cards $1, 6, 41$, and 77.
- The second group contains the cards 7 and 18.
- The third group contains the card 10.