Problem A
Bit String Reordering

Input: Standard Input
Time Limit: 1 second

You have to reorder a given bit string as specified. The only operation allowed is swapping adjacent bit pairs. Please write a program that calculates the minimum number of swaps required.

The initial bit string is simply represented by a sequence of bits, while the target is specified by a *run-length code*. The run-length code of a bit string is a sequence of the lengths of maximal consecutive sequences of zeros or ones in the bit string. For example, the run-length code of "011100" is "1 3 2". Note that there are two different bit strings with the same run-length code, one starting with zero and the other starting with one. The target is either of these two.

In Sample Input 1, bit string "100101" should be reordered so that its run-length code is "1 3 2", which means either "100011" or "011100". At least four swaps are required to obtain "011100". On the other hand, only one swap is required to make "100011". Thus, in this example, 1 is the answer.

Input

The input consists of a single test case. The test case is formatted as follows.

$N\ M$
$b_1\ b_2\ \ldots\ b_N$
$p_1\ p_2\ \cdots\ p_M$

The first line contains two integers N ($1 \leq N \leq 15$) and M ($1 \leq M \leq N$). The second line specifies the initial bit string by N integers. Each integer b_i is either 0 or 1. The third line contains the run-length code, consisting of M integers. Integers p_1 through p_M represent the lengths of consecutive sequences of zeros or ones in the bit string, from left to right. Here, $1 \leq p_j$ for $1 \leq j \leq M$ and $\sum_{j=1}^{M} p_j = N$ hold. It is guaranteed that the initial bit string can be reordered into a bit string with its run-length code $p_1, \ldots, p_M$.

Output

Output the minimum number of swaps required.

Sample Input 1

```
6 3
1 0 0 1 0 1
1 3 2
```

Sample Output 1

```
1
```

Sample Input 2

```
7 2
1 1 1 0 0 0 0
4 3
```

Sample Output 2

```
12
```

Sample Input 3

```
15 14
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
```

Sample Output 3

```
7
```

Sample Input 4

```
1 1
0
1
```

Sample Output 4

```
0
```

Problem B
Miscalculation

Input: Standard Input
Time Limit: 1 second

Bob is an elementary schoolboy, not so good at mathematics. He found Father's calculator and tried cheating on his homework using it. His homework was calculating given expressions containing multiplications and additions. Multiplications should be done prior to additions, of course, but the calculator evaluates the expression from left to right, neglecting the operator precedence. So his answers may be the result of either of the following two calculation rules.

- Doing multiplication before addition
- Doing calculation from left to right neglecting the operator precedence

Write a program that tells which of the rules is applied from an expression and his answer.

An expression consists of integers and operators. All the integers have only one digit, from 0 to 9. There are two kinds of operators `+` and `*`, which represent addition and multiplication, respectively.

The following is an example expression.

```
1+2*3+4
```

Calculating this expression with the multiplication-first rule, the answer is 11, as in Sample Input 1. With the left-to-right rule, however, the answer will be 13 as shown in Sample Input 2.

There may be cases in which both rules lead to the same result and you cannot tell which of the rules is applied. Moreover, Bob sometimes commits miscalculations. When neither rules would result in Bob's answer, it is clear that he actually did.

Input

The input consists of a single test case specified with two lines. The first line contains the expression to be calculated. The number of characters of the expression is always odd and less than or equal to 17. Each of the odd-numbered characters in the expression is a digit from '`0`' to '`9`'. Each of the even-numbered characters is an operator '`+`' or '`*`'. The second line contains an integer which ranges from 0 to 999999999, inclusive. This integer represents Bob's answer for the expression given in the first line.

Output

Output one of the following four characters:

`M` When only the multiplication-first rule results Bob's answer.

`L` When only the left-to-right rule results Bob's answer.

`U` When both of the rules result Bob's answer.

`I` When neither of the rules results Bob's answer.

Sample Input 1	**Sample Output 1**
1+2*3+4 11	M

Sample Input 2	**Sample Output 2**
1+2*3+4 13	L

Sample Input 3	**Sample Output 3**
3 3	U

Sample Input 4	**Sample Output 4**
1+2*3+4 9	I

Problem C
Shopping

Input: Standard Input
Time Limit: 1 second

Your friend will enjoy shopping. She will walk through a mall along a straight street, where N individual shops (numbered from 1 to N) are aligned at regular intervals. Each shop has one door and is located at the one side of the street. The distances between the doors of the adjacent shops are the same length, i.e. a unit length. Starting shopping at the entrance of the mall, she visits shops in order to purchase goods. She has to go to the exit of the mall after shopping.

She requires some restrictions on visiting order of shops. Each of the restrictions indicates that she shall visit a shop before visiting another shop. For example, when she wants to buy a nice dress before choosing heels, she shall visit a boutique before visiting a shoe store. When the boutique is farther than the shoe store, she must pass the shoe store before visiting the boutique, and go back to the shoe store after visiting the boutique.

If only the order of the visiting shops satisfies all the restrictions, she can visit other shops in any order she likes.

Write a program to determine the minimum required walking length for her to move from the entrance to the exit.

Assume that the position of the door of the shop numbered k is k units far from the entrance, where the position of the exit is $N + 1$ units far from the entrance.

Input

The input consists of a single test case.

$$
\begin{array}{l}
N\ m \\
c_1\ d_1 \\
\vdots \\
c_m\ d_m
\end{array}
$$

The first line contains two integers N and m, where N ($1 \leq N \leq 1000$) is the number of shops, and m ($0 \leq m \leq 500$) is the number of restrictions. Each of the next m lines contains two integers c_i and d_i ($1 \leq c_i < d_i \leq N$) indicating the i-th restriction on the visiting order, where she must visit the shop numbered c_i after she visits the shop numbered d_i ($i = 1, \ldots, m$).

There are no pair of j and k that satisfy $c_j = c_k$ and $d_j = d_k$.

Output

Output the minimum required walking length for her to move from the entrance to the exit. You should omit the length of her walk in the insides of shops.

Sample Input 1

```
10 3
3 7
8 9
2 5
```

Sample Output 1

```
23
```

Sample Input 2

```
10 3
8 9
6 7
2 4
```

Sample Output 2

```
19
```

Sample Input 3

```
10 0
```

Sample Output 3

```
11
```

Sample Input 4

```
10 6
6 7
4 5
2 5
6 9
3 5
6 8
```

Sample Output 4

```
23
```

Sample Input 5

```
1000 8
3 4
6 1000
5 1000
7 1000
8 1000
4 1000
9 1000
1 2
```

Sample Output 5

```
2997
```

Problem D
Space Golf

Input: Standard Input
Time Limit: 1 second

You surely have never heard of this new planet surface exploration scheme, as it is being carried out in a project with utmost secrecy. The scheme is expected to cut costs of conventional rover-type mobile explorers considerably, using projected-type equipment nicknamed "observation bullets".

Bullets do not have any active mobile abilities of their own, which is the main reason of their cost-efficiency. Each of the bullets, after being shot out on a launcher given its initial velocity, makes a parabolic trajectory until it touches down. It bounces on the surface and makes another parabolic trajectory. This will be repeated virtually infinitely.

We want each of the bullets to bounce precisely at the respective spot of interest on the planet surface, adjusting its initial velocity. A variety of sensors in the bullet can gather valuable data at this instant of bounce, and send them to the observation base. Although this may sound like a conventional target shooting practice, there are several issues that make the problem more difficult.

- There may be some obstacles between the launcher and the target spot. The obstacles stand upright and are very thin that we can ignore their widths. Once the bullet touches any of the obstacles, we cannot be sure of its trajectory thereafter. So we have to plan launches to avoid these obstacles.
- Launching the bullet almost vertically in a speed high enough, we can easily make it hit the target without touching any of the obstacles, but giving a high initial speed is energy-consuming. Energy is extremely precious in space exploration, and the initial speed of the bullet should be minimized. Making the bullet bounce a number of times may make the bullet reach the target with lower initial speed.
- The bullet should bounce, however, no more than a given number of times. Although the body of the bullet is made strong enough, some of the sensors inside may not stand repetitive shocks. The allowed numbers of bounces vary on the type of the observation bullets.

You are summoned engineering assistance to this project to author a smart program that tells the minimum required initial speed of the bullet to accomplish the mission.

Figure D.1 gives a sketch of a situation, roughly corresponding to the situation of the Sample Input 4 given below.

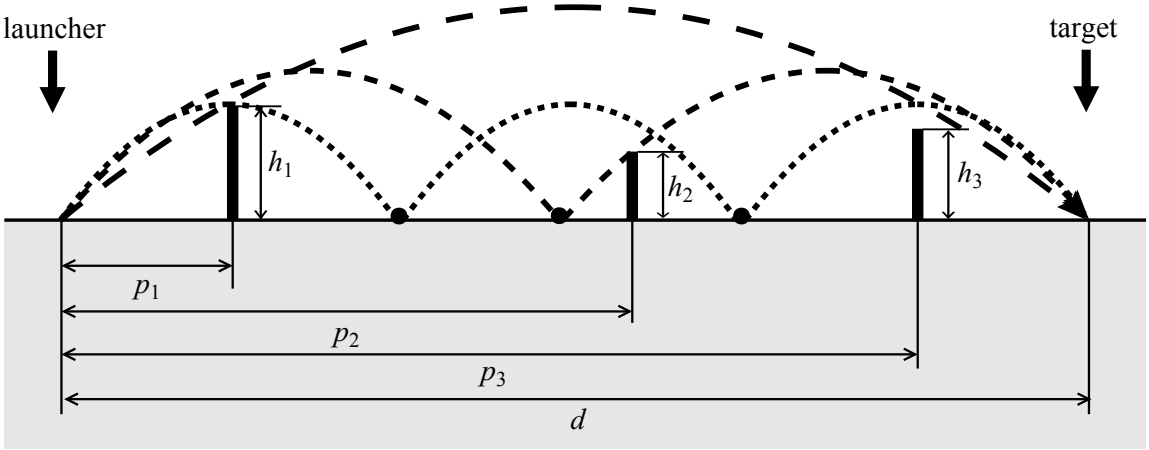

Figure D.1. A sample situation

You can assume the following.

- The atmosphere of the planet is so thin that atmospheric resistance can be ignored.
- The planet is large enough so that its surface can be approximated to be a completely flat plane.
- The gravity acceleration can be approximated to be constant up to the highest points a bullet can reach.

These mean that the bullets fly along a perfect parabolic trajectory.

You can also assume the following.

- The surface of the planet and the bullets are made so hard that bounces can be approximated as elastic collisions. In other words, loss of kinetic energy on bounces can be ignored. As we can also ignore the atmospheric resistance, the velocity of a bullet immediately after a bounce is equal to the velocity immediately after its launch.
- The bullets are made compact enough to ignore their sizes.
- The launcher is also built compact enough to ignore its height.

You, a programming genius, may not be an expert in physics. Let us review basics of rigid-body dynamics.

We will describe here the velocity of the bullet v with its horizontal and vertical components v_{x} and v_{y} (positive meaning upward). The initial velocity has the components v_{ix} and v_{iy}, that is, immediately after the launch of the bullet, $v_{\mathrm{x}} = v_{\mathrm{ix}}$ and $v_{\mathrm{y}} = v_{\mathrm{iy}}$ hold. We denote the horizontal distance of the bullet from the launcher as x and its altitude as y at time t.

- The horizontal velocity component of the bullet is kept constant during its flight when atmospheric resistance is ignored. Thus the horizontal distance from the launcher is proportional to the time elapsed.

$$x = v_{\mathrm{ix}} t \tag{1}$$

- The vertical velocity component v_{y} is gradually decelerated by the gravity. With the gravity acceleration of g, the following differential equation holds during the flight.

$$\frac{dv_{\mathrm{y}}}{dt} = -g \tag{2}$$

  Solving this with the initial conditions of $v_{\mathrm{y}} = v_{\mathrm{iy}}$ and $y = 0$ when $t = 0$, we obtain the following.

$$y = -\frac{1}{2} g t^2 + v_{\mathrm{iy}} t \tag{3}$$

$$= -\left(\frac{1}{2} g t - v_{\mathrm{iy}}\right) t \tag{4}$$

  The equation (4) tells that the bullet reaches the ground again when $t = 2v_{\mathrm{iy}}/g$. Thus, the distance of the point of the bounce from the launcher is $2v_{\mathrm{ix}}v_{\mathrm{iy}}/g$. In other words, to make the bullet fly the distance of l, the two components of the initial velocity should satisfy $2v_{\mathrm{ix}}v_{\mathrm{iy}} = lg$.

- Eliminating the parameter t from the simultaneous equations above, we obtain the following equation that describes the parabolic trajectory of the bullet.

$$y = -\left(\frac{g}{2v_{\mathrm{ix}}^2}\right) x^2 + \left(\frac{v_{\mathrm{iy}}}{v_{\mathrm{ix}}}\right) x \tag{5}$$

For ease of computation, a special unit system is used in this project, according to which the gravity acceleration g of the planet is exactly 1.0.

Input

The input consists of a single test case with the following format.

$d\ n\ b$
$p_1\ h_1$
$p_2\ h_2$
$\vdots$
$p_n\ h_n$

The first line contains three integers, d, n, and b. Here, d is the distance from the launcher to the target spot ($1 \leq d \leq 10000$), n is the number of obstacles ($1 \leq n \leq 10$), and b is the maximum number of bounces allowed, not including the bounce at the target spot ($0 \leq b \leq 15$).

Each of the following n lines has two integers. In the k-th line, p_k is the position of the k-th obstacle, its distance from the launcher, and h_k is its height from the ground level. You can assume that $0 < p_1$, $p_k < p_{k+1}$ for $k = 1, \ldots, n-1$, and $p_n < d$. You can also assume that $1 \leq h_k \leq 10000$ for $k = 1, \ldots, n$.

Output

Output the smallest possible initial speed v_{i} that makes the bullet reach the target. The initial speed v_{i} of the bullet is defined as follows.

$$v_{\mathrm{i}} = \sqrt{{v_{\mathrm{ix}}}^2 + {v_{\mathrm{iy}}}^2}$$

The output should not contain an error greater than 0.0001.

Sample Input 1

```
100 1 0
50 100
```

Sample Output 1

```
14.57738
```

Sample Input 2

```
10 1 0
4 2
```

Sample Output 2

```
3.16228
```

Sample Input 3

```
100 4 3
20 10
30 10
40 10
50 10
```

Sample Output 3

```
7.78175
```

Sample Input 4

```
343 3 2
56 42
190 27
286 34
```

Sample Output 4

```
11.08710
```

Problem E
Automotive Navigation

Input: Standard Input
Time Limit: 5 seconds

The International Commission for Perfect Cars (ICPC) has constructed a city scale test course for advanced driver assistance systems. Your company, namely the Automotive Control Machines (ACM), is appointed by ICPC to make test runs on the course.

The test course consists of streets, each running straight either east-west or north-south. No streets of the test course have dead ends, that is, at each end of a street, it meets another one. There are no grade separated streets either, and so if a pair of orthogonal streets run through the same geographical location, they always meet at a crossing or a junction, where a car can turn from one to the other. No U-turns are allowed on the test course and a car never moves outside of the streets.

Oops! You have just received an error report telling that the GPS (Global Positioning System) unit of a car running on the test course was broken and the driver got lost. Fortunately, however, the odometer and the electronic compass of the car are still alive.

You are requested to write a program to estimate the current location of the car from available information. You have the car's location just before its GPS unit was broken. Also, you can remotely measure the running distance and the direction of the car once every time unit. The measured direction of the car is one of north, east, south, and west. If you measure the direction of the car while it is making a turn, the measurement result can be the direction either before or after the turn. You can assume that the width of each street is zero.

The car's direction when the GPS unit was broken is not known. You should consider every possible direction consistent with the street on which the car was running at that time.

Input

The input consists of a single test case. The first line contains four integers n, x_0, y_0, t, which are the number of streets ($4 \leq n \leq 50$), x- and y-coordinates of the car at time zero, when the GPS unit was broken, and the current time ($1 \leq t \leq 100$), respectively. (x_0, y_0) is of course on some street. This is followed by n lines, each containing four integers x_s, y_s, x_e, y_e describing a street from (x_s, y_s) to (x_e, y_e) where $(x_s, y_s) \neq (x_e, y_e)$. Since each street runs either east-west or north-south, $x_s = x_e$ or $y_s = y_e$ is also satisfied. You can assume that no two parallel streets overlap or meet. In this coordinate system, the x- and y-axes point east and north, respectively. Each input coordinate is non-negative and at most 50. Each of the remaining t lines contains an integer d_i ($1 \leq d_i \leq 10$), specifying the measured running distance from time $i-1$ to i, and a letter c_i, denoting the measured direction of the car at time i and being either `N` for north, `E` for east, `W` for west, or `S` for south.

Output

Output all the possible current locations of the car that are consistent with the measurements. If they are $(x_1, y_1), (x_2, y_2), \ldots, (x_p, y_p)$ in the lexicographic order, that is, $x_i < x_j$ or $x_i = x_j$ and $y_i < y_j$ if $1 \leq i < j \leq p$, output the following:

x_1 y_1
x_2 y_2
$\vdots$
x_p y_p

Each output line should consist of two integers separated by a space.

You can assume that at least one location on a street is consistent with the measurements.

Sample Input 1

```
4 2 1 1
1 1 1 2
2 2 2 1
2 2 1 2
1 1 2 1
9 N
```

Sample Output 1

```
1 1
2 2
```

Sample Input 2

```
6 0 0 2
0 0 2 0
0 1 2 1
0 2 2 2
0 0 0 2
1 0 1 2
2 0 2 2
2 E
7 N
```

Sample Output 2

```
0 1
1 0
1 2
2 1
```

Sample Input 3

```
7 10 0 1
5 0 10 0
8 5 15 5
5 10 15 10
5 0 5 10
8 5 8 10
10 0 10 10
15 5 15 10
10 N
```

Sample Output 3

```
5 5
8 8
10 10
15 5
```

Problem F
There is No Alternative

Input: Standard Input
Time Limit: 3 seconds

ICPC (Isles of Coral Park City) consist of several beautiful islands.

The citizens requested construction of bridges between islands to resolve inconveniences of using boats between islands, and they demand that all the islands should be reachable from any other islands via one or more bridges.

The city mayor selected a number of pairs of islands, and ordered a building company to estimate the costs to build bridges between the pairs. With this estimate, the mayor has to decide the set of bridges to build, minimizing the total construction cost.

However, it is difficult for him to select the most cost-efficient set of bridges among those connecting all the islands. For example, three sets of bridges connect all the islands for the Sample Input 1. The bridges in each set are expressed by bold *edges* in Figure F.1.

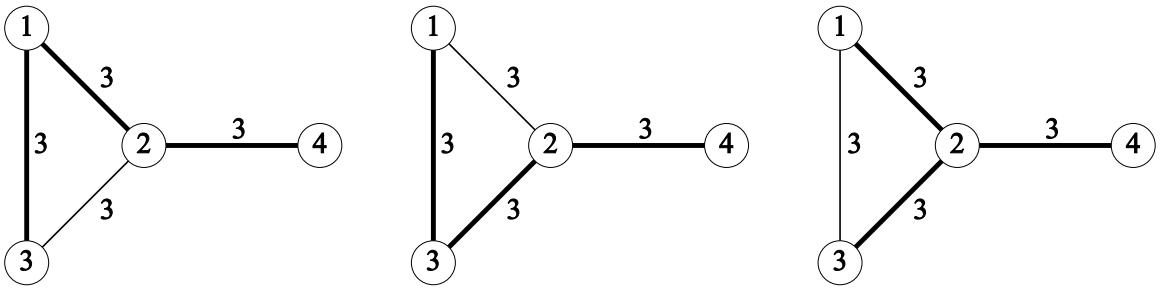

Figure F.1. Three sets of bridges connecting all the islands for Sample Input 1

As the first step, he decided to build only those bridges which are contained in all the sets of bridges to connect all the islands and minimize the cost. We refer to such bridges as *no alternative bridges*. In Figure F.2, no alternative bridges are drawn as thick edges for the Sample Input 1, 2 and 3.

Write a program that advises the mayor which bridges are no alternative bridges for the given input.

Input

The input consists of a single test case.

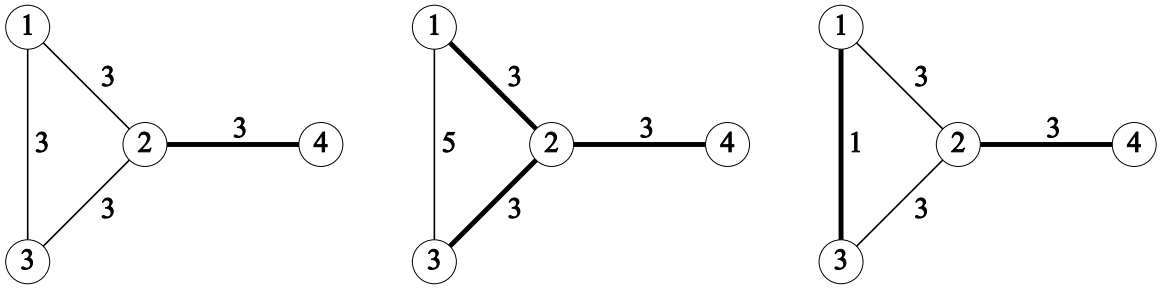

Figure F.2. No alternative bridges for Sample Input 1, 2 and 3

$N\ M$
$S_1\ D_1\ C_1$
$\vdots$
$S_M\ D_M\ C_M$

The first line contains two positive integers N and M. N represents the number of islands and each island is identified by an integer 1 through N. M represents the number of the pairs of islands between which a bridge may be built.

Each line of the next M lines contains three integers S_i, D_i and C_i ($1 \leq i \leq M$) which represent that it will cost C_i to build the bridge between islands S_i and D_i. You may assume $3 \leq N \leq 500$, $N - 1 \leq M \leq \min(50000, N(N-1)/2)$, $1 \leq S_i < D_i \leq N$, and $1 \leq C_i \leq 10000$. No two bridges connect the same pair of two islands, that is, if $i \neq j$ and $S_i = S_j$, then $D_i \neq D_j$. If all the candidate bridges are built, all the islands are reachable from any other islands via one or more bridges.

Output

Output two integers, which mean the number of no alternative bridges and the sum of their construction cost, separated by a space.

Sample Input 1	**Sample Output 1**
4 4 1 2 3 1 3 3 2 3 3 2 4 3	1 3

Sample Input 2

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

Sample Output 2

```
3 9
```

Sample Input 3

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

Sample Output 3

```
2 4
```

Sample Input 4

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

Sample Output 4

```
0 0
```

Problem G
Flipping Parentheses

Input: Standard Input
Time Limit: 5 seconds

A string consisting only of parentheses '(' and ')' is called balanced if it is one of the following.

- A string "()" is balanced.
- Concatenation of two balanced strings are balanced.
- When a string s is balanced, so is the concatenation of three strings "(", s, and ")" in this order.

Note that the condition is stronger than merely the numbers of '(' and ')' are equal. For instance, "())(()" is *not* balanced.

Your task is to keep a string in a balanced state, under a severe condition in which a cosmic ray may flip the direction of parentheses.

You are initially given a balanced string. Each time the direction of a single parenthesis is flipped, your program is notified the position of the changed character in the string. Then, calculate and output the *leftmost* position that, if the parenthesis there is flipped, the whole string gets back to the balanced state. After the string is balanced by changing the parenthesis indicated by your program, next cosmic ray flips another parenthesis, and the steps are repeated several times.

Input

The input consists of a single test case formatted as follows.

$N\ Q$
s
q_1
$\vdots$
q_Q

The first line consists of two integers N and Q ($2 \leq N \leq 300000$, $1 \leq Q \leq 150000$). The second line is a string s of balanced parentheses with length N. Each of the following Q lines is an integer q_i ($1 \leq q_i \leq N$) that indicates that the direction of the q_i-th parenthesis is flipped.

Output

For each event q_i, output the position of the leftmost parenthesis you need to flip in order to get back to the balanced state.

Note that each input flipping event q_i is applied to the string after the previous flip q_{i-1} and its fix.

Sample Input 1	**Sample Output 1**
6 3 ((())) 4 3 1	2 2 1

Sample Input 2	**Sample Output 2**
20 9 ()(((((()))))()()()() 15 20 13 5 3 10 3 17 18	2 20 8 5 3 2 2 3 18

In the first sample, the initial state is "((()))". The 4th parenthesis is flipped and the string becomes "(((())". Then, to keep the balance you should flip the 2nd parenthesis and get ")(())". The next flip of the 3rd parenthesis is applied to the last state and yields "))())". To rebalance it, you have to change the 2nd parenthesis again yielding "(()())".

Problem H
Cornering at Poles

Input: Standard Input
Time Limit: 3 seconds

You are invited to a robot contest. In the contest, you are given a disc-shaped robot that is placed on a flat field. A set of poles are standing on the ground. The robot can move in all directions, but must avoid the poles. However, the robot can make turns around the poles touching them.

Your mission is to find the shortest path of the robot to reach the given goal position. The length of the path is defined by the moving distance of the center of the robot. Figure H.1 shows the shortest path for a sample layout. In this figure, a red line connecting a pole pair means that the distance between the poles is shorter than the diameter of the robot and the robot cannot go through between them.

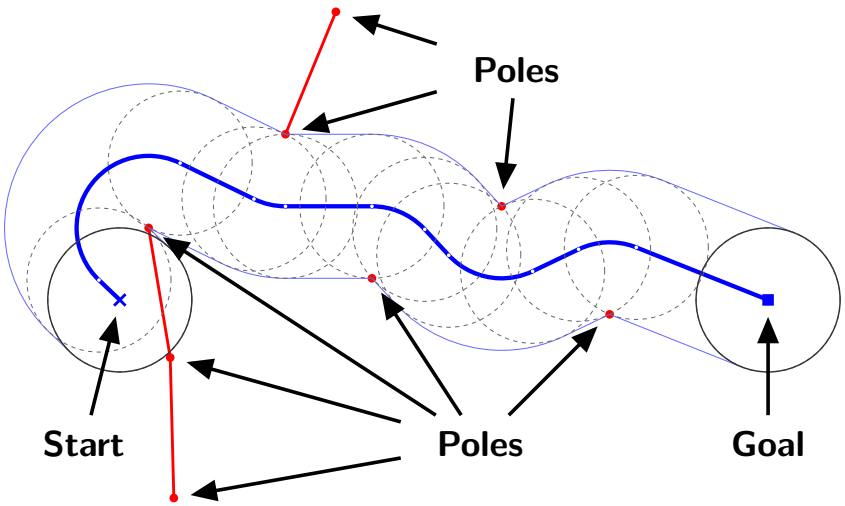

Figure H.1. The shortest path for a sample layout

Input

The input consists of a single test case.

$N\ G_x\ G_y$
$x_1\ y_1$
$\vdots$
$x_N\ y_N$

The first line contains three integers. N represents the number of the poles ($1 \leq N \leq 8$). (G_x, G_y) represents the goal position. The robot starts with its center at (0, 0), and the robot accomplishes its task when the center of the robot reaches the position (G_x, G_y). You can assume that the starting and goal positions are not the same.

Each of the following N lines contains two integers. (x_i, y_i) represents the standing position of the i-th pole. Each input coordinate of (G_x, G_y) and (x_i, y_i) is between -1000 and 1000, inclusive. The radius of the robot is 100, and you can ignore the thickness of the poles. No pole is standing within a 100.01 radius from the starting or goal positions. For the distance $d_{i,j}$ between the i-th and j-th poles ($i \neq j$), you can assume $1 \leq d_{i,j} < 199.99$ or $200.01 < d_{i,j}$.

Figure H.1 shows the shortest path for Sample Input 1 below, and Figure H.2 shows the shortest paths for the remaining Sample Inputs.

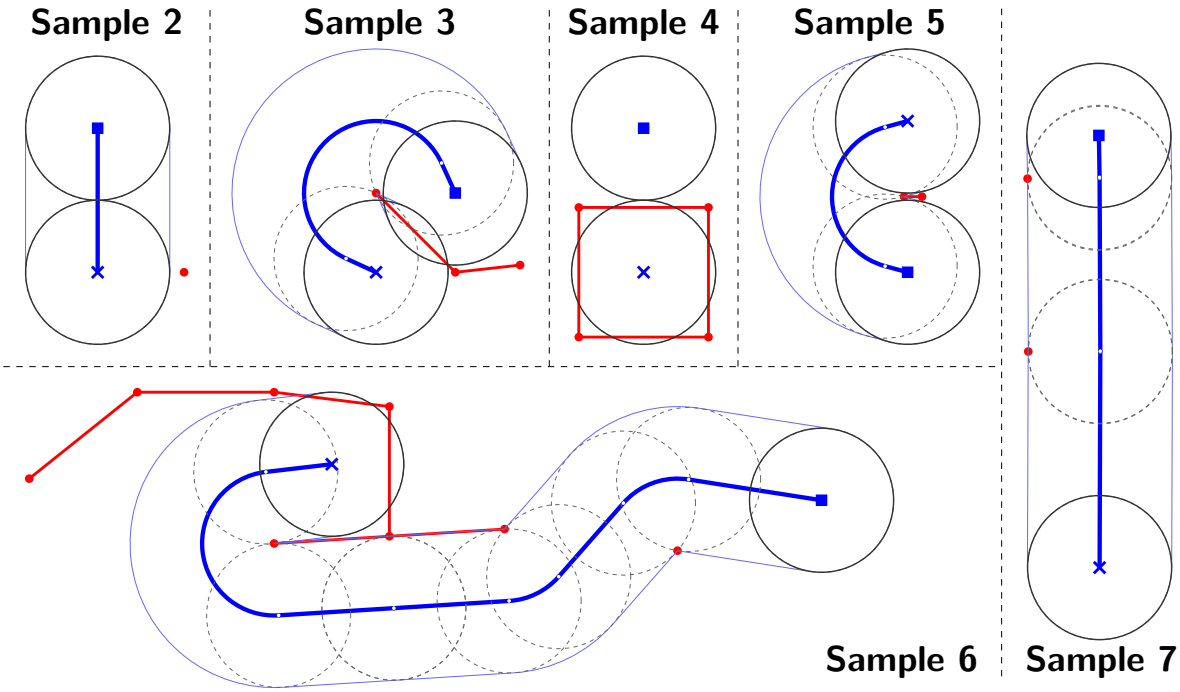

Figure H.2. The shortest paths for the sample layouts

Output

Output the length of the shortest path to reach the goal. If the robot cannot reach the goal, output 0.0. The output should not contain an error greater than 0.0001.

Sample Input 1

```
8 900 0
40 100
70 -80
350 30
680 -20
230 230
300 400
530 130
75 -275
```

Sample Output 1

```
1210.99416
```

Sample Input 2

```
1 0 200
120 0
```

Sample Output 2

```
200.0
```

Sample Input 3

```
3 110 110
0 110
110 0
200 10
```

Sample Output 3

```
476.95048
```

Sample Input 4

```
4 0 200
90 90
-90 90
-90 -90
90 -90
```

Sample Output 4

```
0.0
```

Sample Input 5

```
2 0 -210
20 -105
-5 -105
```

Sample Output 5

```
325.81116
```

Sample Input 6

```
8 680 -50
80 80
80 -100
480 -120
-80 -110
240 -90
-80 100
-270 100
-420 -20
```

Sample Output 6

```
1223.53071
```

Sample Input 7	Sample Output 7
2 -1 600 -99 300 -100 540	600.01216

Problem I
Sweet War

Input: Standard Input
Time Limit: 1 second

There are two countries, Imperial Cacao and Principality of Cocoa. Two girls, Alice (the Empress of Cacao) and Brianna (the Princess of Cocoa) are friends and both of them love chocolate very much.

One day, Alice found a transparent tube filled with chocolate balls (Figure I.1). The tube has only one opening at its top end. The tube is narrow, and the chocolate balls are put in a line. Chocolate balls are identified by integers 1, 2, ..., N where N is the number of chocolate balls. Chocolate ball 1 is at the top and is next to the opening of the tube. Chocolate ball 2 is next to chocolate ball 1, ..., and chocolate ball N is at the bottom end of the tube. The chocolate balls can be only taken out from the opening, and therefore the chocolate balls must be taken out in the increasing order of their numbers.

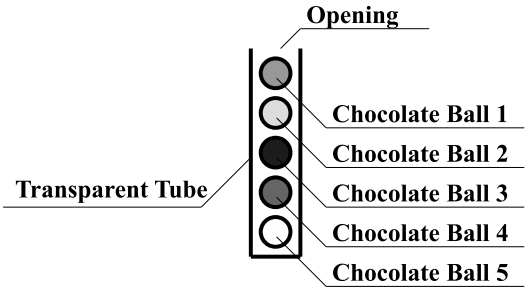

Figure I.1. Transparent tube filled with chocolate balls

Alice visited Brianna to share the tube and eat the chocolate balls together. They looked at the chocolate balls carefully, and estimated that the *nutrition value* and the *deliciousness* of chocolate ball i are r_i and s_i, respectively. Here, each of the girls wants to maximize the sum of the deliciousness of chocolate balls that she would eat. They are sufficiently wise to resolve this conflict peacefully, so they have decided to play a game, and eat the chocolate balls according to the rule of the game as follows:

1. Alice and Brianna have initial energy levels, denoted by nonnegative integers A and B, respectively.
2. Alice and Brianna takes one of the following two actions in turn:
   - **Pass**: she does not eat any chocolate balls. She gets a little hungry – specifically, her energy level is decreased by 1. She cannot pass when her energy level is 0.

- **Eat**: she eats the topmost chocolate ball – let this chocolate ball i (that is, the chocolate ball with the smallest number at that time). Her energy level is increased by r_i, the nutrition value of chocolate ball i (and NOT decreased by 1). Of course, chocolate ball i is removed from the tube.

3. Alice takes her turn first.
4. The game ends when all chocolate balls are eaten.

You are a member of the staff serving for Empress Alice. Your task is to calculate the sums of deliciousness that each of Alice and Brianna can gain, when both of them play optimally.

Input

The input consists of a single test case. The test case is formatted as follows.

$$
\begin{array}{l}
N\ A\ B \\
r_1\ s_1 \\
r_2\ s_2 \\
\vdots \\
r_N\ s_N
\end{array}
$$

The first line contains three integers, N, A and B. N represents the number of chocolate balls. A and B represent the initial energy levels of Alice and Brianna, respectively. The following N lines describe the chocolate balls in the tube. The chocolate balls are numbered from 1 to N, and each of the lines contains two integers, r_i and s_i for $1 \leq i \leq N$. r_i and s_i represent the nutrition value and the deliciousness of chocolate ball i, respectively. The input satisfies

- $1 \leq N \leq 150$,
- $0 \leq A, B, r_i \leq 10^9$,
- $0 \leq s_i$, and
- $\displaystyle\sum_{i=1}^{N} s_i \leq 150$.

Output

Output two integers that represent the total deliciousness that Alice and Brianna can obtain when they play optimally.

Sample Input 1	**Sample Output 1**
2 5 4 5 7 4 8	8 7

Sample Input 2

```
3 50 1
49 1
0 10
0 1
```

Sample Output 2

```
10 2
```

Sample Input 3

```
4 3 2
1 5
2 46
92 40
1 31
```

Sample Output 3

```
77 45
```

Sample Input 4

```
5 2 5
56 2
22 73
2 2
1 55
14 18
```

Sample Output 4

```
57 93
```

Problem J
Exhibition

Input: Standard Input
Time Limit: 10 seconds

The city government is planning an exhibition and collecting industrial products. There are n candidates of industrial products for the exhibition, and the city government decided to choose k of them. They want to minimize the total price of the k products. However, other criteria such as their sizes and weights should be also taken into account. To address this issue, the city government decided to use the following strategy: The i-th product has three parameters, price, size, and weight, denoted by x_i, y_i, and z_i, respectively. Then, the city government chooses k items $i_1, \ldots, i_k$ $(1 \leq i_j \leq n)$ that minimizes the evaluation measure

$$e = \Big(\sum_{j=1}^{k} x_{i_j}\Big)\Big(\sum_{j=1}^{k} y_{i_j}\Big)\Big(\sum_{j=1}^{k} z_{i_j}\Big),$$

which incorporates the prices, the sizes, and the weights of products. If there are two or more choices that minimize e, the government chooses one of them uniformly at random.

You are working for the company that makes the product 1. The company has decided to cut the price, reduce the size, and/or reduce the weight of the product so that the city government may choose the product of your company, that is, to make the probability of choosing the product positive. We have three parameters A, B, and C. If we want to reduce the price, size, and weight to $(1-\alpha)x_1$, $(1-\beta)y_1$, and $(1-\gamma)z_1$ $(0 \leq \alpha, \beta, \gamma \leq 1)$, respectively, then we have to invest $\alpha A + \beta B + \gamma C$ million yen. We note that the resulting price, size, and weight do not have to be integers. Your task is to compute the minimum investment required to make the city government possibly choose the product of your company. You may assume that employees of all the other companies are too lazy to make such an effort.

Input

The input consists of a single test case with the following format.

$$\begin{array}{l} n\ k\ A\ B\ C \\ x_1\ y_1\ z_1 \\ x_2\ y_2\ z_2 \\ \vdots \\ x_n\ y_n\ z_n \end{array}$$

The first line contains five integers. The integer n $(1 \leq n \leq 50)$ is the number of industrial products, and the integer k $(1 \leq k \leq n)$ is the number of products that will be chosen by the

city government. The integers A, B, C ($1 \leq A, B, C \leq 100$) are the parameters that determine the cost of changing the price, the size, and the weight, respectively, of the product 1.

Each of the following n lines contains three integers, x_i, y_i, and z_i ($1 \leq x_i, y_i, z_i \leq 100$), which are the price, the size, and the weight, respectively, of the i-th product.

Output

Output the minimum cost (in million yen) in order to make the city government possibly choose the product of your company. The output should not contain an absolute error greater than 10^{-4}.

Sample Input 1

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

Sample Output 1

```
0.631579
```

Sample Input 2

```
6 1 1 2 3
10 20 30
1 2 3
2 4 6
3 6 9
4 8 12
5 10 15
```

Sample Output 2

```
0.999000
```

Problem K

L_∞ Jumps

Input: Standard Input
Time Limit: 3 seconds

Given two points (p, q) and (p', q') in the XY-plane, the L_∞ *distance* between them is defined as $\max(|p - p'|, |q - q'|)$. In this problem, you are given four integers n, d, s, t. Suppose that you are initially standing at point $(0, 0)$ and you need to move to point (s, t). For this purpose, you perform jumps exactly n times. In each jump, you must move exactly d in the L_∞ distance measure. In addition, the point you reach by a jump must be a lattice point in the XY-plane. That is, when you are standing at point (p, q), you can move to a new point (p', q') by a single jump if p' and q' are integers and $\max(|p - p'|, |q - q'|) = d$ holds.

Note that you cannot stop jumping even if you reach the destination point (s, t) before you perform the jumps n times.

To make the problem more interesting, suppose that some cost occurs for each jump. You are given $2n$ additional integers $x_1, y_1, x_2, y_2, \ldots, x_n, y_n$ such that $\max(|x_i|, |y_i|) = d$ holds for each $1 \leq i \leq n$. The cost of the i-th (1-indexed) jump is defined as follows: Let (p, q) be a point at which you are standing before the i-th jump. Consider a set of lattice points that you can jump to. Note that this set consists of all the lattice points on the edge of a certain square. We assign integer 1 to point $(p + x_i, q + y_i)$. Then, we assign integers $2, 3, \ldots, 8d$ to the remaining points in the set in the counter-clockwise order. (Here, assume that the right direction is positive in the x-axis and the upper direction is positive in the y-axis.) These integers represent the costs when you perform a jump to these points.

For example, Figure K.1 illustrates the points reachable by your i-th jump when $d = 2$ and you are at $(3, 1)$. The numbers represent the costs for $x_i = -1$ and $y_i = -2$.

Compute and output the minimum required sum of the costs for the objective.

Input

The input consists of a single test case.

$n\ d\ s\ t$
$x_1\ y_1$
$x_2\ y_2$
$\vdots$
$x_n\ y_n$

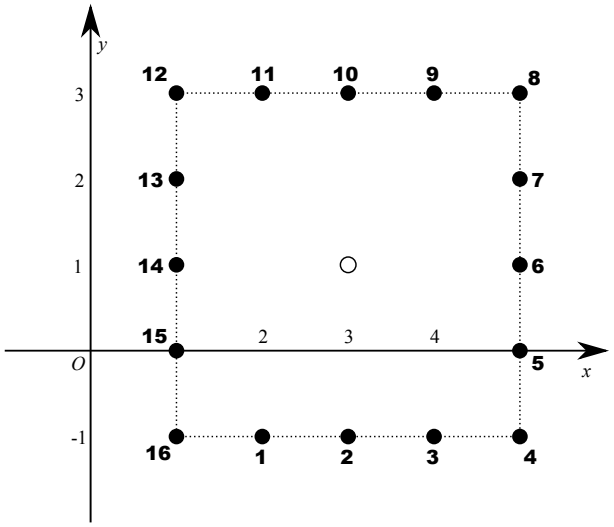

Figure K.1. Reachable points and their costs

The first line contains four integers. n ($1 \leq n \leq 40$) is the number of jumps to perform. d ($1 \leq d \leq 10^{10}$) is the L_∞ distance which you must move by a single jump. s and t ($|s|, |t| \leq nd$) are the x and y coordinates of the destination point. It is guaranteed that there is at least one way to reach the destination point by performing n jumps.

Each of the following n lines contains two integers, x_i and y_i with $\max(|x_i|, |y_i|) = d$.

Output

Output the minimum required cost to reach the destination point.

Sample Input 1

```
3 2 4 0
2 2
-2 -2
-2 2
```

Sample Output 1

```
15
```

Sample Input 2

```
4 1 2 -2
1 -1
1 0
1 1
-1 0
```

Sample Output 2

```
10
```

Sample Input 3

```
6 5 0 2
5 2
-5 2
5 0
-3 5
-5 4
5 5
```

Sample Output 3

```
24
```

Sample Input 4

```
5 91 -218 -351
91 91
91 91
91 91
91 91
91 91
```

Sample Output 4

```
1958
```

Sample Input 5

```
1 10000000000 -10000000000 -2527532346
8198855077 10000000000
```

Sample Output 5

```
30726387424
```