2017 Team Formation Test

Task Overview

ID	Name	Time Limit	Memory Limit	Subtasks
T171	Optimal Bowing	1.000 s	256 MB	14 + 30 + 29 + 27
T172	City Reform	1.000 s	256 MB	13 + 12 + 19 + 15 + 41
T173	Secret Meeting	0.500 s	512 MB	15 + 21 + 20 + 18 + 16 + 10
T174	Constellation	Output-Only		10 + 10 + 10 + 10 + 10 + 10 + 10 + 10 + 10 + 10

Notice:
Unless otherwise specified, inputs and outputs shall follow the format below:

- One space between a number and another number or character in the same line.
- No space between characters in the same line.
- Each string shall be placed in its own separate line.
- Outputs will be automatically fixed as follows: Trailing spaces in each line will be removed and an end-of-line character will be added to the end of the output if not present. All other format errors will not be fixed.

C++ programmers should be aware that using C++ streams (`cin` / `cout`) may lead to I/O bottlenecks and substantially lower performance.

For some problems 64-bit integers may be required. In Pascal it is `int64`. In C/C++ it is `long long` and its token for `scanf` /`printf` is `%lld`.

All tasks are divided into subtasks. You need to pass all test cases in a subtask to get points.

T171 - OPTIMAL BOWING

Time Limit: 1.000 s / Memory Limit: 256 MB

Alice is going to perform violin solo in the school hall. The hall is very large. She must play loud enough so that even the audiences sitting far from the stage can also enjoy her performance.

The music piece that Alice is going to play consists of N music notes. It is described by N integers $a_1, a_2, \ldots, a_N$ where $1 \leq a_i \leq 10$. The integer a_i represents the duration of the i^{th} note in the music, and the duration of that note is $\frac{1}{a_i}$ of the duration of a whole note. For example, $\boxed{1}$ represents a whole note and $\boxed{4}$ represents a quarter note.

A violin is played by drawing the bow back and forth across a string. When the bow is drawn from its bottom to its tip, it is called a down-bow; when the bow is drawn from its tip to its bottom, it is called an up-bow. The faster the bow moves, the louder the sound will be. For example, playing a whole note with the whole bow (e.g. draw from the bottom of the bow to the tip of the bow) is louder than playing a whole note with half the bow (e.g. draw from the bottom of the bow to the middle of the bow).

Playing every note with the whole bow is of course the loudest, but it is not desirable because the notes will be played with uneven loudness. For example, a quarter note will be louder than a whole note when both are played with the whole bow. As a perfectionist violinist, Alice wants to play every note with the same loudness. Thus, she always plays a note with the amount of bow proportional to the duration of that note. For example, if she plays whole notes with length L of the bow, she will only play quarter notes with length $\frac{L}{4}$ of the bow, so that they will have the same loudness.

Alice has not decided the bowing yet, i.e. whether each note should be played with a down-bow or up-bow. A bowing is a sequence of N symbols, each symbol is either $\boxed{u}$ (up-bow) or $\boxed{d}$ (down-bow). Alice is experienced enough that she knows what the initial position of the bow should be and how much the bow should move once she is given the bowing. Please help her to design an optimal bowing that allows her to play as loud as possible while each note is played with the same loudness.

INPUT

The first line contains an integer N, the number of notes in the music piece.

The second line contains N integers $a_1, a_2, \ldots, a_N$, the duration of each note.

OUTPUT AND SCORING

Output one line containing N characters $\boxed{d}$ and $\boxed{u}$, denoting an optimal bowing that gives the maximum loudness.

You score 100% if your optimal bowing is the lexicographically smallest one. Here, $\boxed{d}$ is considered to be smaller than $\boxed{u}$. You score 80% if you output any other bowing that also gives the maximum loudness.

SAMPLE TESTS

	Input	Output
1	3 2 1 2	dud

The initial position is the middle of the bow

	Input	Output
2	5 1 9 8 10 1	duduu

The initial position is the bottom of the bow

SUBTASKS

For all cases: $1 \leq a_i \leq 10$, $1 \leq N \leq 2000$

	Points	Constraints
1	14	$1 \leq a_i \leq 2$
2	30	$1 \leq a_i \leq 7$ $1 \leq N \leq 18$
3	29	$1 \leq a_i \leq 7$ $1 \leq N \leq 100$
4	27	No additional constraints

T172 - CITY REFORM

Time Limit: 1.000 s / Memory Limit: 256 MB

Hackerland has N cities and $N-1$ roads. The i^{th} road connects cities a_i and b_i and has length l_i $(1 \le l_i \le 5)$. All pairs of cities are connected by exactly one path.

Now King RB wants to reform the roads of Hackerland. He wants to form a so-called Regional Business OPeration zone, which should consist of a cycle. To do so in the most economic way, RB would like to add exactly one road to the city network, connecting two different cities. All the cities connected by the cycle are considered to be a part of the RBOP zone.

Now, for a particular way of reforming the city network (by adding road (u, v)), define its inconvenience $I(u, v)$ to be the maximum distance of any city from the RBOP zone.

Clearly, it is better to minimize inconvenience, but RB's not-so-clever secretary AT had no idea how to do so :(To save himself from embarrassment, he generated (perhaps randomly!) Q ways of adding one road, and asked his subordinates to help calculate, for each of the Q ways (u_i, v_i), the value of $I(u_i, v_i)$. AT hoped that, by doing so and picking the plan with minimal $I(u_i, v_i)$, he can prepare an acceptable plan for the city reform.

You happen to be one of AT's subordinates. Calculate the values for him.

INPUT

The first line of the input contains two integers N and Q, the number of cities and plans.

Then $N - 1 + Q$ lines follow.

The first $N-1$ of these lines contain three integers a_i, b_i and l_i each, the i^{th} road connects city a_i and b_i, and its length is l_i.

The last Q of these lines contain two integers u_i and v_i each, which are the plans generated by AT.

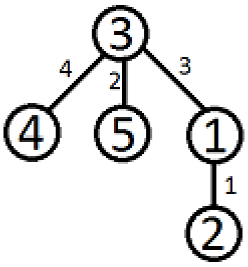
Note that $u_i \neq v_i$, but (u_i, v_i) may be an existing road.

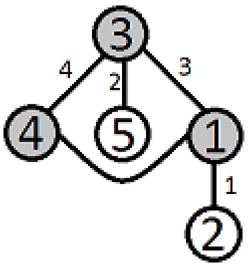
OUTPUT

On the i^{th} line, output $I(u_i, v_i)$, the inconvenience of the i^{th} plan.

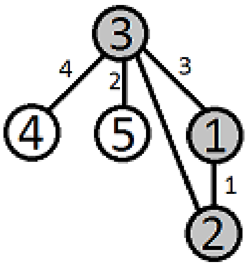
SAMPLE TESTS

	Input	Output
1	5 3	2
	1 2 1	4
	1 3 3	7
	3 4 4	
	3 5 2	
	1 4	
	2 3	
	1 2	

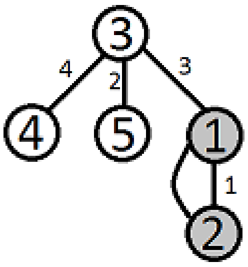
Original map : 

Map for plan 1 (the gray cities are inside RBOP zone) : 

The farthest city is city 5, distance is 2.

Map for plan 2 (the gray cities are inside RBOP zone) : 

The farthest city is city 4, distance is 4.

Map for plan 3 (the gray cities are inside RBOP zone) : 

The farthest city is city 4, distance is 7.

2

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

```
8
0
7
5
```

3

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

```
4
5
4
5
```

4

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

```
4
5
3
7
```

SUBTASKS

For all cases: $2 \leq N \leq 200000, 1 \leq Q \leq 200000$

	Points	**Constraints**
1	13	The i^{th} road connects city i and city $i+1$
2	12	The i^{th} road connects city 1 and city $i+1$
3	19	$Q \leq 50$, $N \leq 50$
4	15	$Q \leq 5000$, $N \leq 5000$
5	41	No additional constraints

T173 - SECRET MEETING

Time Limit: 0.500 s / Memory Limit: 512 MB

Manhotan is a gigantic city that resembles a grid on the Cartesian coordinate plane. Its border can be regarded as a square on the Cartesian plane, with opposite vertices $(-B, -B)$ and (B, B), where $B = 2 \times 10^9$. Its metro system is the best in the world. For every $-B \leq i \leq B$, there is an east-west railway connecting $(-B, i)$ and (B, i) and a north-south railway connecting $(i, -B)$ and (i, B).

Citizens of Manhotan (called Manhotanians) live in intersections of roads. Their positions can be regarded as lattice points (x, y) on the coordinate plane.

Manhotan metro has rather strange regulations. Let C and D be positive constants. If a person starts from (x_1, y_1) and wishes to go to (x_2, y_2), he/she must first travel along the east-west direction to reach the point (x_2, y_1) with cost $C \times |x_1 - x_2|$, then travel along the north-south direction to reach the destination with an additional cost $D \times |y_1 - y_2|$. This, Manhotan officials claim, can "minimize load on the metro system".

Thus, for any person, the travel cost from (x_1, y_1) to (x_2, y_2) is $C \times |x_1 - x_2| + D \times |y_1 - y_2|$. Manhotanians, however, all have the Nonapus card and can use it to travel the cheaper of the two trips (if exists) for free. Hence, the travel cost for Nonapus card holders is $max(C \times |x_1 - x_2|, D \times |y_1 - y_2|)$.

Today is day 0. N Manhotanians just met each other online, forming a group called Truly Fantastic Thrillers (TFT). As the name suggests, they wanted to produce high-quality thrillers, which are currently banned in Manhotan due, according to officials, to "its negative effects on personal health and on society as a whole".

Now on day 0, the i-th person is positioned at (x_i, y_i). They have to be cautious to avoid arrests, so on every day the i-th person will move his/her position by the vector (u_i, v_i). In other words, his/her position on day t is $(x_i + t \times u_i, y_i + t \times v_i)$. It is possible that $u_i = v_i = 0$, in which case the person moves out briefly, then moves in again, perhaps to a different flat.

Note that in Manhotan, only points with both x- and y-coordinates even are residential areas. Therefore, x_i, y_i, u_i, v_i **are all even**.

They would like to hold a meeting at some place (X, Y) (X and Y have to be integers, but not necessarily even), on one of the next T days. Concretely, they would like to choose a day t_{opt}, where $1 \leq t_{opt} \leq T$, and a meeting place (X, Y) which minimizes the maximum cost that one of the members has to pay to travel to (X, Y) on day t_{opt}. If there are several possible days, they will prefer the earliest one.

Later today, the government will announce a reform on metro pricing system, setting the constants C and D, effective immediately. To plan ahead, the members make Q guesses about the two constants. The i-th guess is (C_i, D_i). For each guess, they want to find:

- the minimum cost for the N members to meet at a common place on one of the next T days, and
- the earliest possible day which gives this minimum cost.

INPUT

The first line of input consists of two integers N and T.
For the next N lines, the i-th line consists of four even integers x_i, y_i, u_i, v_i, where (x_i, y_i) is the position of the i-th person **on day 0** and (u_i, v_i) is the movement vector of the i-th person.
The next line of input consists of an integer Q.
For the next Q lines, the i-th line consists of two integers C_i, D_i, parameters of the i-th guess on the metro pricing system, as described.

OUTPUT

Output Q lines in total.
The i-th line of the output should contain two integers. The first integer should be the earliest day to achieve the minimized maximum cost for all N members to attend the meeting, and the second integer should be that cost.
You are reminded that day 0 will not be considered for the meeting.

SAMPLE TESTS

Input **Output**

1

```
3 4
-10 -2 6 2
2 4 2 -2
2 8 2 0
3
1 1
10 3
7 3
```

```
1 4
3 15
2 14
```

Their positions at $t = 1$: $(-4, 0), (4, 2), (4, 8)$
Their positions at $t = 2$: $(2, 2), (6, 0), (6, 8)$
Their positions at $t = 3$: $(8, 4), (8, -2), (8, 8)$
Their positions at $t = 4$: $(14, 6), (10, -4), (10, 8)$
For the first query, they should meet on day 1 at $(0, 4)$, with maximum cost 4.
For the second query, they should meet on day 3 at $(8, 3)$, with maximum cost 15.
For the third query, they should meet on day 2 at $(4, 4)$, with maximum cost 14.

2

```
2 2
0 0 0 0
6 0 -4 0
1
1 1
```

```
1 1
```

Both $t = 1$ and $t = 2$ give the same maximum cost 1, so choose the earliest time $(t = 1)$.

SUBTASKS

For all cases: $1 \le N \le 4 \times 10^4$, $1 \le T \le 5 \times 10^5$, $-10^9 \le x_i, y_i \le 10^9$, $-2000 \le u_i, v_i \le 2000$, $1 \le Q \le 10^5$, $1 \le C_i, D_i \le 10^9$.
In addition, x_i, y_i, u_i, v_i are all even numbers.
Note that the constraints guarantee that they will not go beyond the boundary of the city during the T days.

	Points	**Constraints**
1	15	$N \le 50$ $T = 1$ $Q \le 10$ $-200 \le x_i + u_i \le 200$, $-200 \le y_i + v_i \le 200$
2	21	$1 \le N \le 50$ $1 \le T \le 200$ $1 \le Q \le 2000$
3	20	$1 \le N \le 50$ $1 \le Q \le 2000$
4	18	$1 \le T \le 1000$ $1 \le Q \le 2000$
5	16	$1 \le T \le 30000$ $1 \le Q \le 2000$
6	10	No additional constraints

T174 - CONSTELLATION

Output-Only

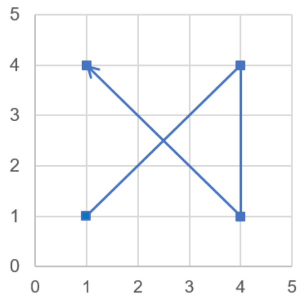
While lying in the bed, Alice looks at the numerous stars in the sky. She thinks that constellations are silly for various reasons:

- There are many ways to connect the stars to form shapes. Alice thinks that the constellations are just some random shapes.
- The distances between the stars within a constellation can be large.

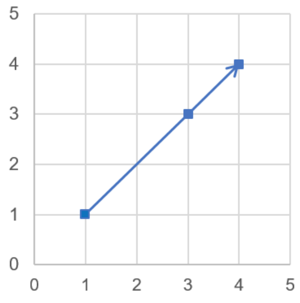
Suddenly Alice has an idea -- is there a scientific way to form constellations?

She defines the problem as follows: Assume that each of the N stars in the sky correspond to a point in a 2D Cartesian coordinate plane. The stars have **integral** coordinates (x_i, y_i) and no 2 stars have the same coordinates. Now let's draw a polyline (broken line) made of V vertices to cover the stars. The only rules is that the vertices must have integral coordinates between -50 and 50 inclusive. It is not required to turn only where there is a star. Concretely, the following scenarios are allowed:

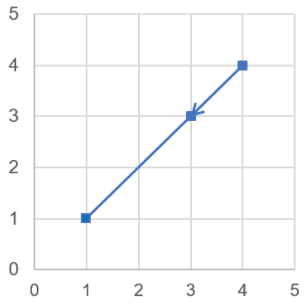
Self intersection:
$(1,1) \to (4,4) \to (4,1) \to (1,4)$



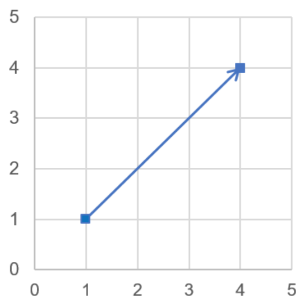
No turn:
$(1,1) \to (3,3) \to (4,4)$



180 degree turn:
$(1,1) \to (4,4) \to (3,3)$



Staying at same point:
$(1,1) \to (4,4) \to (4,4)$



Alice wants her new constellation to cover as many stars as possible. A star only counts once even if the polyline passes through the star two or more times. Can you do this for her?

This is an output-only task consisting of 10 independent test cases in separate input files. You are only required to submit the outputs that corresponds to the given input files.

INPUT

The first line of each case, contains 3 integers: N -- the number of stars, V -- the number of vertices in the polyline and T -- the target number of stars to cover.

Each of the next N lines contains 2 integers x_i and y_i, the coordinates of the stars. $(-40 \leq x_i, y_i, \leq 40)$

No two stars have the same coordinates. For your convenience, the coordinates have been sorted by x, then y.

OUTPUT

Output V lines, the coordinates of vertices of the constellation (polyline) in order. Each line should contain two integers x_i and y_i. ($-50 \le x_i, y_i \le 50$).

To read from file and write to file, you may either use file I/O or input redirection: `./programname < starsX.txt > constX.txt`

During contest, it is also possible to submit (upload) a single output file instead of a zip file. The filename must match one of the output filenames (case-sensitive).

Output-Only Task Information

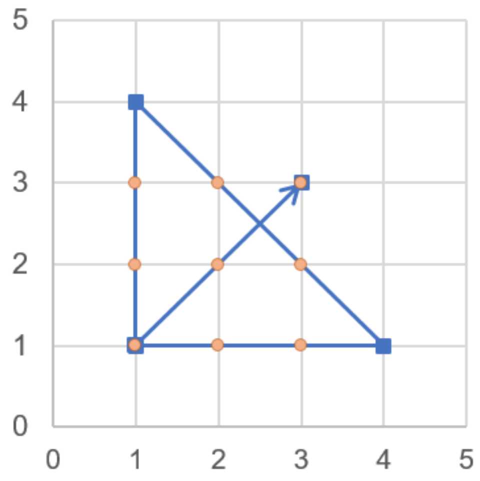
This is an output-only task. Please place all the output files in the root of a zip file. Do not place them in a subfolder. Windows/*nix line endings will be corrected automatically. Maximum size of the zip file is 2^{24} bytes (16MB).

Output filename requirements: `const1.txt` `const2.txt` `const3.txt` `const4.txt` `const5.txt` `const6.txt` `const7.txt` `const8.txt` `const9.txt` `const10.txt`

SAMPLE TESTS

	Input	Output

1	9 5 9
	1 1
	1 2
	1 3
	2 1
	2 2
	2 3
	3 1
	3 2
	3 3
		1 1
		1 4
		4 1
		1 1
		3 3



SCORING

It is obvious that a polyline consisting of V vertices can cover at least V points. Therefore, if your constellation covers fewer than V stars, you score 0% for the case.

On the other hand, if your polyline covers more than T stars, you score 100% for the case. It is guaranteed that there exists a solution that covers at least T stars. However, it is not guaranteed that there exists a solution that covers all N stars.

Otherwise, if your polyline covers P points, you will score $10 \times 10^{(P-V)/(T-V)}$ percent. Therefore if you cover exactly V stars, you score $10 \times 10^0 = 10$ percent.

TEST CASE OVERVIEW

Case	Input	Output	N	V	T
1	`stars1.txt`	`const1.txt`	25	2	10
2	`stars2.txt`	`const2.txt`	49	13	49
3	`stars3.txt`	`const3.txt`	12	7	12
4	`stars4.txt`	`const4.txt`	80	3	18
5	`stars5.txt`	`const5.txt`	200	41	180
6	`stars6.txt`	`const6.txt`	90	20	90
7	`stars7.txt`	`const7.txt`	40	11	28
8	`stars8.txt`	`const8.txt`	120	25	115
9	`stars9.txt`	`const9.txt`	200	35	185
10	`stars10.txt`	`const10.txt`	200	50	200