Problem A. Death Star

Input file:	`deathstar.in`
Output file:	`deathstar.out`
Time limit:	2 seconds
Memory limit:	64 mebibytes

A long time ago in a galaxy far, far away The Merchant Federation fought against the rebels — a small group of defenders of the Republic. For the ultimate victory the Federation has decided to build the Death Star — a huge space ship, a weapon that the Universe had never seen before. They selected several planets and build a factory on each one so that all together they became a big circular conveyor: the details produced at the planet a_i were transported to the planet number a_{i+1} for $i < M$ and from the planet a_M to the planet number a_1 (naturally, all M planets are distinct). To somewhat reduce the cost of the project, the Federation also decided to transport the details only through the existing hypertunnels, each of which connects some two planets in a certain direction.

Having heard of the Death Star, the rebels realized that the war would not last for too long yet: either the Death Star would destroy them or they would destroy it (and the Federation as well). It's not very hard to understand that the Republic defenders consider satisfying only the second alternative...

The Analysis Center took the decision to destroy the Death Star at the construction stage. So they need to find out which planets were selected by the Federation. They already have the hypertunnel map. It's also known that for the years of tyranny, the Federation had managed to build exactly one hypertunnel among each pair of planets. In order to find out whether there is a factory on the planet or not, they had to send there a scout ship (the rebels really had very few of them). For this reason, first of all they needed to consider the matter theoretically, and only after that send scouts to the plantes which, as they thought, contained the factories most probably.

Actually, the most urgent question is to count how many different ways of selecting the locations for the factories and the transportation routes the Federation had. And since you worked in the Analysis Center, this task was assigned exactly to you.

Two ways are considered the same if their canonic forms do not differ. The canonic form is a sequence of M numbers $a_j, a_{j+1}, \ldots, a_M, a_1, \ldots, a_{j-1}$ such that a_j is minimal among all a_k and there are tunnels $a_j \to a_{j+1}, a_{j+1} \to a_{j+2}, \ldots, a_{M-1} \to a_M, a_M \to a_1, \ldots, a_{j-1} \to a_j$.

Input

On the first line of the input file there are two integer numbers: N — the number of planets in the Galaxy, M — the number of the factories the Federation intends to build ($1 \le N \le 1400$, $2 < M < 6$). On the second line of the input file there is a sequence of hexidecimal digits (`0..9`, `A..F`) which specifies the hypertunnel map in the ciphered way. In order to decipher it, one should to replace each digit with its binary representation (keep in mind that if the latter is shorter than 4 digits, it is padded with zeroes to 4 digits at the left end). Then one should take only first $N * (N-1)/2$ digits and the rest of the sequence must be dropped.

The hypertunnel description is given in the following order: $(1-2)$, $(1-3)$, ..., $(1-n)$, $(2-3)$, $(2-4)$, ..., $(2-n)$, ..., $((n-1)-n)$, where each pair $(a-b)$ has exactly one corresponding digit which is 1 when the tunnel from a to b exists and 0 otherwise (then there exists a hypertunnel from b to a).

Output

Print one integer — the number of ways to select the locations for the factories. And may The Force be with you!

Example

`deathstar.in`	`deathstar.out`

`4 3`
`B4`
	`2`

Comment on the sample:

After decipering the string "`B4`" we obtain the string "`10110100`". Dropping off 3 unnecessary trailing digits, we see that there are tunnels $1 \to 2$, $3 \to 1$, $1 \to 4$, $2 \to 3$, $4 \to 2$, $3 \to 4$. And the required ways are: $1 \to 2 \to 3$, $2 \to 3 \to 4$.

Problem B. Elevator

Input file:	`elevator.in`
Output file:	`elevator.out`
Time limit:	2 seconds
Memory limit:	64 mebibytes

Let's consider some building with N floors. There is an elevator in it, of course. It works in the following way:

1. if the button is pressed on some floor, the number of this floor is pushed into the calls queue;
2. if at the moment when button is pressed, elevator is not moving and the moving has not taken place or no less than A seconds have passed since the moment elevator stopped, then the elevator starts moving to the floor at which the button was pressed. This call will be the current call;
3. it takes B seconds for the elevator to move between consecutive floors;
4. if the elevator passes a floor, number of which is in the calls queue, it stops immediately (we will consider this moment as the moment when the corresponding call is processed);
5. after the stop, one occurrence of floor's number is removed from the calls queue, lift stays motionless during A seconds (time for passengers to enter and leave), and then continues moving to the floor of the current call;
6. if the elevator stops at the floor of the current call, then the next earliest call becomes the current one;
7. if the call from the current floor occurs at the moment when the elevator starts moving, the elevator immediately stops again.

You are required to determine the moments of time when the calls will be processed. Initially the elevator is on the first floor.

Input

On the first line of the input file there are 4 integer numbers: N — the number of floors, M — the number of calls, A — the number of seconds for passengers to enter and leave the elevator, B — the number of seconds for the elevator to move between two consecutive floors, ($1 \le N, M \le 10^5$, $1 \le A, B \le 100$). The next M lines contain 2 integer numbers each: T_i — the number of seconds passed since the time moment 0 till the i-th call was made, F_i — the number of floor from which the i-th call was made ($1 \le T_i \le 10^5$, $1 \le F_i \le N$, $T_i \le T_{i+1}$ for all i in range $1..M-1$). In case when $T_i = T_{i+1}$, the i-th call is considered to be made earlier than the $(i+1)$-th call. The numbers on each line are separated by spaces.

Output

Output the moments when the calls were processed (the number of seconds passed since the moment 0 till the moment when the call was processed) in the order the calls were given in the input file, one per line.

Example

elevator.in	elevator.out

5 6 12 13
42 5
86 1
154 4
183 3
234 2
234 4
	94
	158
	221
	196
	272
	234

Problem C. Illumination

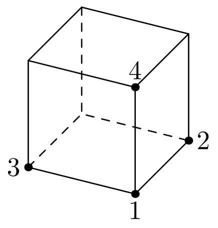
Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	64 mebibytes

Two cubes and a light bulb are placed in a three-dimensional euclidean space. You are expected to find out if one of them casts shadow on the other one and if so, calculate the area of this shadow.

Input

The first line of the input contains the coordinates of the bulb. It is followed by two groups of four lines each that describe the cubes. Each line of the cube description contains the coordinates of a vertex (see the figure where the vertices are marked and labeled in the same order as they are given in the input).

All input numbers are not greater than 10^4 by their absolute values.



All the coordinates are given with 5 digits after decimal point. It is guaranteed that the cubes do not intersect, the light bulb is outside both of them, and doesn't lie on any of the planes that contain their faces. A light bulb should be regarded as a point light source.

Output

The output should contain a single line with two numbers separated with a space character. The first one is the number of the cube that has a shadow on it (1 or 2). The second is the area of the shadow with 3 digits after decimal point.

If none of the given cubes casts shadow on the other the output should contain a single number "-1".

Example

standard input	standard output

`-1.00000 1.00000 1.00000`
`0.00000 0.00000 0.00000`
`2.00000 0.00000 0.00000`
`0.00000 2.00000 0.00000`
`0.00000 0.00000 2.00000`
`5.00000 0.00000 0.00000`
`7.00000 0.00000 0.00000`
`5.00000 2.00000 0.00000`
`5.00000 0.00000 2.00000`
	`2 4.000`

`0.00000 0.00000 0.00000`
`1.00000 1.00000 1.00000`
`2.00000 1.00000 1.00000`
`1.00000 2.00000 1.00000`
`1.00000 1.00000 2.00000`
`-1.00000 -1.00000 -1.00000`
`-1.00000 -2.00000 -1.00000`
`-2.00000 -1.00000 -1.00000`
`-1.00000 -1.00000 -2.00000`
	`-1`

Problem D. Road Works

Input file:	`rw.in`
Output file:	`rw.out`
Time limit:	3 seconds
Memory limit:	256 mebibytes

In a far land of Roadland, there are N cities connected by M bidirectional roads. The road network is organized in such a way that

- no road connects a city to itself,
- no two cities are connected by more than one road,
- each city can be reached from each other city by roads, and
- if cities A and B are connected by road, there exists at most one simple path from A to B which does not contain that road (Recall that a *simple* path visits each vertex not more than once).

Each road is described by which cities it connects, as well as what number of seconds it takes initially to drive along that road.

Starting from day 1 and until day K, a single event happens every day. There are two types of events. The first type is a query of the form "what is the minimal time to get from city x to city y". The second type is an information message of the form "the road number x was repaired this day, and it now takes y seconds to drive along that road". All roads suffer damage over time, so at the beginning of each day, before any event takes place, the time to drive along each road is increased by one second.

Given the road network and the sequence of events, find the answers of the queries.

Input

The first line of input contains two integers N and M ($2 \le N \le 10^5$).

Next M lines contain three integers each: a_i, b_i and t_i. Here, a_i and b_i are the numbers of cities that the respective road connects, and t_i is the time it takes to drive along that road ($1 \le a_i, b_i \le N$, $1 \le t_i \le 10^5$). Roads are numbered 1 through M in the order in which they are given.

The next line of input contains a single integer K ($1 \le K \le 10^5$) — the number of events.

Next K lines contain the descriptions of events, one line per event. A query is given as `P` x y ($1 \le x, y \le N$), and an information message is given as `R` x y ($1 \le x \le M$, $1 \le y \le 10^5$). All numbers in the input are integers.

Output

For each query, output a single line with a single integer on it — the minimal time it takes to drive between the cities given in the query.

Example

rw.in	rw.out
4 4 1 2 1 2 3 1 3 4 1 4 2 1 10 P 1 2 P 2 3 P 1 4 P 1 3 R 1 5 R 2 1000 P 1 2 P 2 4 R 2 1 P 1 4	2 3 8 10 7 9 21