Problem A. Bitonic Travelling Byteman

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	2 seconds
Memory limit:	256 mebibytes

There are n cities in Byteland, numbered from 1 to n. Some pairs of cities are connected by bidirectional roads of various lengths. Travelling Byteman would like to design a route over Byteland, visiting each city exactly once and going back to the starting point of the route. Additionally Byteman has one more requirement: he would like subsequent cities of the route to form a bitonic sequence, that is, a sequence $m_1, m_2, \ldots, m_n$ satisfying $m_1 < m_2 < \ldots < m_i > m_{i+1} > \ldots > m_n$, where i is arbitrary, in particular one might have $i = n$ (then we have an increasing sequence) or $i = 1$ (then we have a decreasing sequence).

Your task is to help Byteman in designing his route of shortest length, or state that no such route exists.

Input

The first line of input contains two integers n and m ($2 \le n \le 200\,000$, $1 \le m \le 500\,000$). Each of the next m lines contains a description of a road in Byteland. Each road is described by 3 positive integers a_i, b_i, w_i ($a_i \le b_i \le n$, $a_i \ne b_i$, $w_i \le 10^9$), meaning that there exists a road connecting cities a_i and b_i of length w_i. Each pair of cities is connected by at most one bidirectional road.

Output

The single line of output should contain a single word „`NIE`" (i.e., *no* in Polish), if the required route does not exist, or the length of the shortest route that Byteman is looking for.

Examples

Standard input	Standard output

`4 5`
`1 2 2`
`1 3 3`
`2 3 3`
`3 4 4`
`1 4 1`
	`10`

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

Problem B. Chessboard (Division 1 Only!)

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Byteman admires chess puzzles and he is a long-time subscriber of the Chess Player's Magazine. The most recent issue of this magazine contains the following puzzle:

> You are given a chessboard of size $n \times n$. In how many ways can you emplace n rooks on the chessboard, so that no two rooks attack each other and the i-th rook is located neither in the i-th column nor in the i-th row? The rooks, the rows and the columns are numbered from 1 to n. The result should be given modulo m.

Puzzles of the type "mate in 13 moves" are a piece of cake for Byteman, but this new type of puzzle seems very hard for him. Could you please help him?

Input

The only line of input contains two integers n and m ($1 \le n \le 10^{18}$, $1 \le m \le 10^6$).

Output

Your program should output the number of possible placement of rooks.

Example

Standard input	Standard output
`3 120`	`4`

Problem C. Desks

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	3 seconds
Memory limit:	**32 mebibytes**

There are n students in a class, where n is an even number. Some students like each other and some do not. However if a student x likes a student y, then for sure the student y likes the student x. A teacher would like to pair all the students so that in each of the $n/2$ desks in the classroom sit exactly two students that like each other. Your task is to calculate the number of ways the teacher can pair the students assuming that desks are indistinguishable.

Input

The first line of the input contains exactly two integers n $(2 \le n \le 26)$ and m $(0 \le m \le n(n-1)/2)$. Each of the following m lines contains exactly two integers $1 \le a, b \le n$ and denotes the fact that students a and b like each other. You may assume that n is even and that each pair $\{a, b\}$ appears in the input at most once.

Output

In the first line of the output your program should write the number of possible ways the teacher can make students sit in the classroom.

Example

Standard input	Standard output

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

Problem D. Difference

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	2 seconds
Memory limit:	256 mebibytes

A word consisting of n lower-case letters of the English alphabet ('a'–'z') is given. We would like to choose a non-empty contiguous (i.e. one-piece) fragment of the word so as to maximise the difference in the number of occurrences of the most and the least frequent letter in the fragment. We are assuming that the least frequent letter has to occur at least once in the resulting fragment. In particular, should the fragment contain occurrences of only one letter, then the most and the least frequent letter in it coincide.

Input

The first line of the standard input holds one integer n ($1 \le n \le 1\,000\,000$) that denotes the length of the word. The second line holds a word consisting of n lower-case letters of the English alphabet.

Output

The first and only line of the standard output is to hold a single integer, equal to the maximum difference in the number of occurrences of the most and the least frequent letter that is attained in some non-empty contiguous fragment of the input word.

Example

Standard input	Standard output
10 aabbaaabab	3

Explanation of the example: The fragment that attains the difference of 3 in the number of occurrences of `a` and `b` is `aaaba`.

Problem E. Garbage

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	9 seconds
Memory limit:	256 mebibytes

The Byteotian Waste Management Company (BWMC) has drastically raised the price of garbage collection lately. This caused some of the citizens to stop paying for collecting their garbage and start disposing of it in the streets. Consequently, many streets of Byteburg are literally buried under litter.

The street system of Byteburg consists of n intersections, some of which are directly connected with bidirectional streets. No two streets connect the same pair of intersections. Some of the streets are littered while others are not.

The mayor of Byteburg, Byteasar, has decided on an unprecedented action to persuade the citizens to pay for waste collection. Namely, he decided to clean only some of the streets — precisely those that the majority of people living on paid for garbage collection. The streets that the majority of people living on did not pay for waste collection, on the other hand, will thus remain littered — or if it is called for — will become littered by the garbage collected from other streets! Byteasar has already prepared a city map with the streets to be cleaned and to remain or become littered marked on. Unfortunately, the BWMC employees cannot comprehend his master plan. They are, however, quite capable of carrying out simple instructions.

A single such instruction consists in driving the garbage truck along a route that starts on an arbitrary intersection, goes along any streets of choice, and ending on the very same intersection that it started on. However, every intersection can be visited at most once on a single route, except for the one it starts and ends with—the garbage truck obviously appears twice on that one. The truck cleans a littered street it rides along, but on the other hand it dumps the waste on the clean streets along its route, making them littered.

Byteasar wonders if it is possible to execute his plan by issuing a number of aforementioned route instructions. Help him out by writing a program that determines a set of such routes or concludes that it is impossible.

Input

There are two integers, separated by a single space, in the first line of the standard input: n and m ($1 \le n \le 10^5$, $1 \le m \le 10^6$), denoting the number of intersections and the number of streets in Byteburg, respectively. The intersections are numbered from 1 to n. The following m lines specify successive streets, one per line. Each of those lines holds four integers separated by single spaces: a, b, s and t ($1 \le a < b \le n$, $s, t \in \{0, 1\}$). Such a quadruple specifies that the intersections a and b are connected with a street, s tells if the street is currently littered (0 means clean, while 1 means littered), and t tells if the street should be littered according to Byteasar's plan.

You may assume that if there exists a set of routes to carry out Byteasar's plan, then there is one in which the total number of streets that the garbage truck follows does not exceed $5 \cdot m$.

Output

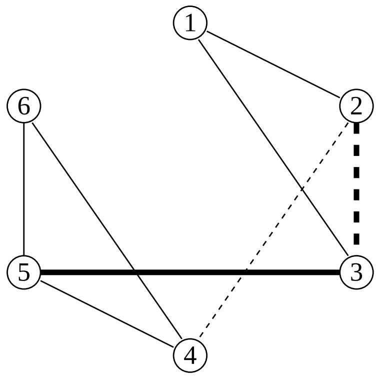
If there is no set of routes for the garbage truck to execute Byteasar's plan, then the word "`NIE`" (*no* in Polish) should be printed out to the first and only line of the standard output. Otherwise, an arbitrary set of routes that does execute Byteasar's plan and has the truck move along no more than $5 \cdot m$ streets in total should be printed out. Then the first line should hold the number k of routes in the set. The following k lines should describe the routes, one per line. The $(i + 1)$-th line should start with a positive integer k_i equal to the number of streets in the i-th route. After a single space, k_i+1 numbers of successive intersections along the route should follow, separated by single spaces.

Examples

The clean streets are drawn with a thin line in the figure, whereas the littered streets are drawn with a thick line. The streets that are to be clean are drawn with a dashed line, while those that are to be littered are drawn with a solid line.

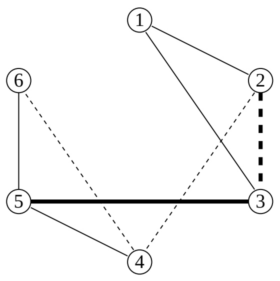
Standard input	Standard output

6 8
1 2 0 1
2 3 1 0
1 3 0 1
2 4 0 0
3 5 1 1
4 5 0 1
5 6 0 1
4 6 0 1
	2
	3 1 3 2 1
	3 4 6 5 4



Standard input	Standard output

6 8
1 2 0 1
2 3 1 0
1 3 0 1
2 4 0 0
3 5 1 1
4 5 0 1
5 6 0 1
4 6 0 0
	NIE



Problem F. Highways (Division 1 Only!)

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	12 seconds
Memory limit:	256 mebibytes

Byteland is a small country, where n cities are connected by $n-1$ bidirectional regular roads. From each city one can travel to every other city using regular roads only, which causes severe traffic jams. For this reason several highways have been built, where each highway goes between some pair of cities.

By a route we mean a sequence of regular roads and/or highways, connecting adjacent cities. All cities on a route are distinct. For each pair of cities x, y there exists exactly one route which does not use any highway; we call such a route the main route between x and y.

People going from a city x to a city y can either choose the main route, or use some highway. In the latter case the route cannot intersect the main route except for the cities x and y and must contain exactly one highway.

Your task is to calculate the number of routes people can take between given pairs of cities (cities in one pair are distinct).

Input

The first line of input contains a single integer n ($1 \le n \le 10^5$) — the number of cities in Byteland. Cities are numbered from 1 to n. Each of the next $n-1$ lines contains two integers a_i, b_i ($1 \le a_i \ne b_i \le n$) meaning that cities a_i and b_i are connected by a regular road.

The next line contains an integer m ($1 \le m \le 10^5$) — the number of highways. Each of the next m lines contains two integers a_i, b_i ($1 \le a_i \ne b_i \le n$) meaning that cities a_i and b_i are connected by a highway.

The next line contains an integer q ($1 \le q \le 5 \cdot 10^5$) — the number of queries. Each of the next q lines contains a description of a query.

Queries are given in the same format as regular roads or highways.

Output

Your program should output exactly q lines. The i-th line should contain the number of routes for the i-th query.

Examples

Standard input	Standard output

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

Problem G. Imprisoning Godzilla (Division 1 Only!)

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	6 seconds
Memory limit:	256 mebibytes

The citizens of Bytetown finally managed to capture the insolent monster Godzilla! Now they encountered a problem of where to jail the monster…

One of the citizens finally got an idea to imprison it using a fence of triangular shape, with vertices coinciding with three of the trees of Byteforest. Obviously, the best option is to minimize the area of this triangle, so that the part of the forest free of this dangerous creature is as large as possible. Your task is to find out how large must this triangular area be.

Input

The first line of the input contains an integer n $(3 \le n \le 2\,000)$. Each of the following n lines contains two integers x_i, y_i $(-10^9 \le x_i, y_i \le 10^9)$ which denote the coordinates of a tree in Byteforest. No two trees grow in the same point of the forest. Moreover, not all the trees are collinear.

Output

Your program should output one number denoting **twice** the area of the smallest possible triangle. We are not interested in zero-area triangles formed of three collinear points, since one cannot keep Godzilla inside a fence of such a shape.

Example

Standard input	Standard output

5
-7 4
-7 2
7 -2
-5 5
5 -4
	4

Problem H. Tree Rotations (Division 1 Only!)

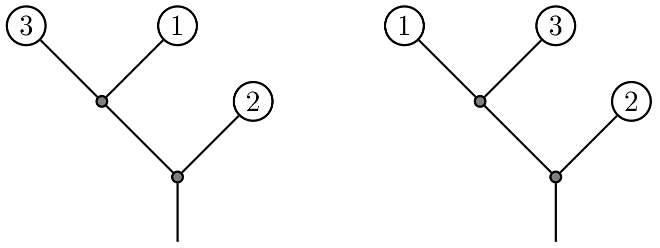
Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	12 seconds
Memory limit:	**128 mebibytes**

Byteasar the gardener is growing a rare tree called *Rotatus Informatikus*. It has some interesting features:

- The tree consists of straight branches, bifurcations and leaves. The trunk stemming from the ground is also a branch.
- Each branch ends with either a bifurcation or a leaf on its top end.
- Exactly two branches fork out from a bifurcation at the end of a branch — the left branch and the right branch.
- Each leaf of the tree is labelled with an integer from the range $1..n$. The labels of leaves are unique.
- With some gardening work, a so called *rotation* can be performed on any bifurcation, swapping the left and right branches that fork out of it.

The corona of the tree is the sequence of integers obtained by reading the leaves' labels from left to right.

Byteasar is from the old town of Byteburg and, like all true Byteburgers, praises neatness and order. He wonders how neat can his tree become thanks to appropriate rotations. The neatness of a tree is measured by the number of *inversions* in its corona, i.e. the number of pairs (i, j), $1 \le i < j \le n$ such that $a_i > a_j$ in the corona $a_1, a_2, \ldots, a_n$.



The original tree (on the left) with corona $3, 1, 2$ has two inversions. A single rotation gives a tree (on the right) with corona $1, 3, 2$, which has only one inversion. Each of these two trees has 5 branches.

Write a program that determines the minimum number of inversions in the corona of Byteasar's tree that can be obtained by rotations.

Input

In the first line of the standard input there is a single integer n ($2 \le n \le 1\,000\,000$) that denotes the number of leaves in Byteasar's tree. Next, the description of the tree follows. The tree is defined recursively:

- if there is a leaf labelled with p ($1 \le p \le n$) at the end of the trunk (i.e., the branch from which the tree stems), then the tree's description consists of a single line containing a single integer p,
- if there is a bifurcation at the end of the trunk, then the tree's description consists of three parts:
  - the first line holds a single number 0,

- then the description of the left subtree follows (as if the left branch forking out of the bifurcation was its trunk),
- and finally the description of the right subtree follows (as if the right branch forking out of the bifurcation was its trunk).

Output

In the first and only line of the standard output a single integer is to be printed: the minimum number of inversions in the corona of the input tree that can be obtained by a sequence of rotations.

Example

Standard input	Standard output
3 0 0 3 1 2	1

The figure illustrates the tree given in the example.

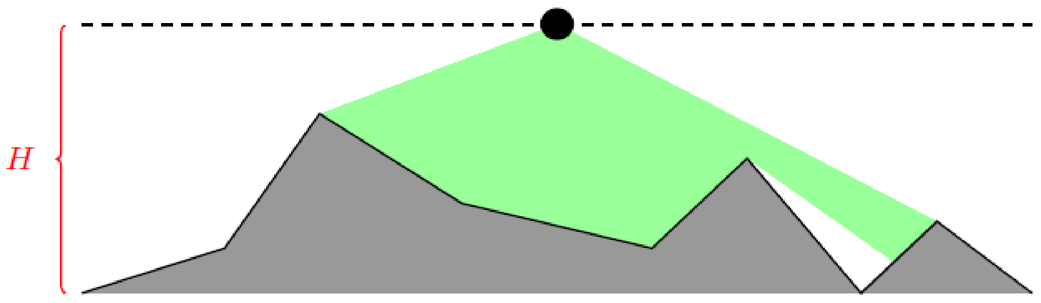
Problem I. Mountains

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Martial law was recently introduced in Byteland. The military government is trying to eliminate the remaining oppositional activists. This is not that simple, though.

Thanks to (not really voluntary) specialists' reports the government managed to figure out several possible locations where the activists might have thier main base. Now the authorities decided to wait until the activists blow thier own cover. For this, all the possible locations of the base will be watched using several spy satellites.

One could imagine Byteland as a two-dimensional mountain landscape obtained by connecting several points with different x coordinates:



Spy satellites can only be placed on the line $y = H$. Each such satellite is able to watch all points that can be connected with the satellite's position by a line segment that does not intersect the polyline formed by the landscape (however, the line segment may touch the polyline). Your task is to count the minimal number of satellites which are necessary to simultaneously watch all potential locations of the activists' base.

Input

The first line of the input contains two integers n and H ($1 \le n \le 100\,000$, $1 \le H \le 1\,000\,000$). The following n lines contain a description of the landscape. Each of those lines contains three integers x_i, y_i and z_i ($0 \le x_i \le 1\,000\,000$, $0 \le y_i < H$, $z_i \in \{0, 1\}$). Here (x_i, y_i) denotes the coordinates of the point; $z_i = 1$ means that this point is a possible location of the activists' base, otherwise $z_i = 0$. You may assume that y_1 and y_n are always equal 0 and that all points are given in a strictly increasing order of the x-coordinate.

Output

The first and only line of the output should contain a single integer — the minimal number of satellites needed.

Example

Standard input	Standard output

9 30
0 0 1
15 5 1
25 20 1
40 10 1
60 5 1
70 15 1
82 0 1
90 8 1
100 0 1
	2

Explanation of the example. It suffices to place two satellites in $x = 31.(6)$ and in $x = 112$.

Problem J. Strongbox (Division 1 Only!)

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Byteasar is a famous safe-cracker, who renounced his criminal activity and got into testing and certifying anti-burglary devices. He has just received a new kind of strongbox for tests: a combinatorial safe. A combinatorial safe is something different from a combination safe, even though it is opened with a rotary dial. The dial can be set in n different positions, numbered from 0 to $n-1$. Setting the dial in some of these positions opens the safe, while in others it does not. And here is the combinatorial property, from which the name comes from: if x and y are opening positions, then so is $(x+y) \bmod n$ too; note that is holds for $x = y$ as well.

Byteasar tried k different positions of the dial: $m_1, m_2, \ldots, m_k$. The positions $m_1, m_2, \ldots, m_{k-1}$ did not open the safe, only the last position m_k did. Byteasar is already tired from checking these k positions and has thus absolutely no intention of trying the remaining ones. He would like to know however, based on what he already knows about the positions he tried, what is the maximum possible number of positions that open the safe. Help him by writing an appropriate program!

Input

The first line of the standard input gives two integers n and k, separated by a single space, $1 \le k \le 250\,000$, $k \le n \le 10^{14}$. The second line holds k different integers, also separated by single spaces, $m_1, m_2, \ldots, m_k$, $0 \le m_i < n$. You can assume that the input data correspond to a certain combinatorial safe that complies with the description above.

Output

Your program should print out to the first and only line of the standard output a single integer: the maximum number of the dial's positions that can open the safe.

Example

Standard input	Standard output

`42 5`
`28 31 10 38 24`
	`14`

Problem K. Circles (Division 2 Only!)

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

There is a large circle with radius R and n small circles with radius r that are placed inside on the border of the large circle. Each small circle touch large circle and two (one, if $n = 2$) other small circles. Your job is, given R and n, to compute r.

Input

First line of input file contains one integer T ($1 \leq T \leq 4100$) — number of test cases. Each test case consists of one line containing a float R (not more than six digits after decimal point) and an integer n ($1 \leq R \leq 100$, $2 \leq n \leq 100$).

Output

For each test case print in new line radius r of small circle with precision 10^{-3}.

Example

Standard input	Standard output

`4`
`5.0 7`
`5.0 19`
`3.1415 20`
`56 2`
	`1.513`
	`0.707`
	`0.425`
	`28.000`

Problem L. Notebook delivery (Division 2 Only!)

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

For winners of the programming contest, driver need to deliver X notebooks, while the car can at most take Y notebooks.

If there are more, than Y notebooks, the car is will to break down due to the weight. If there are less, than X, that means driver has not fulfilled the request.

At the shop, there is a row of packages he can bring along. Each package contains a specified number of notebooks.

Because lack of time due of road traffic, driver decided to choose one entirely random starting point in the row of packages. Then he chooses a random end point among the packages from the starting point to the end of row, inclusive. All the packs between those points, inclusive, are loaded onto the car.

What is the actual probability for each of three cases (first — not enough notebooks to fulfill the request, second — all is OK, maybe some notebooks are left after delivery, third — breaking of car) to occur? Assume that the driver will always be able to fit all the packs into the car and both the starting point and the ending point of are chosen with a uniform probability distribution.

Input

Input file consists of 3 lines. First of them contains a single integer N ($1 \le N \le 2 \cdot 10^5$) — the number of packages in the shop. The second line contains a sequence $B_1, \ldots, B_N$ of N characters between '`A`' and '`Z`' each — amount of notebooks in each of N packages, in order as they are located in the shop. '`A`' represents an empty package, '`B`' — a package with one notebook inside etc, so '`Z`' represents a package with 25 notebooks in it. The last line consists of two integers L and U. L is the number of notebooks the driver is supposed to deliver, while U is the maximum number of notebooks the car can take before it will break down ($0 \le L \le U \le 5 \cdot 10^4$).

Output

Print 3 numbers on a single line. The first number gives the probability that the driver succeeds, the second one that that he brings along too few notebooks, and the third one that the car breaks down. Answers within 10^{-6} of the precise answer are accepted.

Examples

Standard input
4 ZDHA 2 10
Standard output
0.5 0.25 0.25

Standard input
3 DEF 5 6
Standard output
0.3333333333 0.2777777778 0.3888888889

Problem M. Grapping Hooks (Division 2 Only!)

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Once in Middle-Earth human and elven armies together had trying to defeat orcish castle.

To attack a castle next tactics has used: soldiers would line up neatly in front of the castle's wall and throw their grappling hooks over the walls. If one does not throw straight it can easily happen that two hooks cross, making it impossible for the two warriors to climb the wall. That is why every king made his soldiers practice a lot so that this would not happen in combat.

Due to their perfect training the hooks will never cross within an army (human or elven), however it can happen that a hook thrown by a human crosses one thrown by a elf.

Given the positions of human and elven soldiers as well as the positions they threw their grappling hooks at, determine how many distinct pairs of human and elven soldiers crossed their hooks.

If there are n humans and m elves, the positions in the line where the soldiers are standing are numbered from 1 to $n + m$. The positions on the castle's wall are numbered from 1 to $n + m$ as well, where position i is directly opposite of position i on the line the soldiers are standing on. A grappling hook thrown from position i to j is said to cross another hook thrown from k to l if and only if ($i < k$ and $j \geq l$) or ($i > k$ and $j \leq l$).

Grappling hooks of the same race will never cross each other, nor will two soldiers occupy the same position on the line. However, two hooks (of different races) can be thrown to the same position in which case they are said to cross each other as well.

Input

First line of input file consists of two integers n and m, the number of human and elven soldiers, respectively ($1 \leq n, m \leq 3 \cdot 10^4$). The next n lines describe the human soldiers followed by m more lines for the elven soldiers. Each line consists of two integer i and j separated by a space, indicating the soldier's position i and the position j he threw his grappling hook to ($1 \leq i, j \leq n + m$).

Output

Print the number of distinct pairs of soldiers whose grappling hooks are crossed.

Examples

Standard input	Standard output

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

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

Problem N. 4 Letters (Division 2 Only!)

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

You are given a list of strings over the alphabet '`X`', '`Y`', '`Z`, '`W`', and your task is to find the shortest string (which is typically not listed) that contains all given strings as sub-strings. If there are several such strings of shortest length, find the smallest in lexicographical order.

Input

First line of the input file contains the number n of strings ($1 \leq n \leq 15$). Then follow these strings, one per line. Each string is not empty, not longer that 100 symbols and consists of the letters '`X`', '`Y`', '`Z`, '`W`' only.

Output

Print a line containing the shortest (and smallest) string as described in statement.

Example

Standard input	Standard output
2 ZXYWYW YWZ	ZXYWYWZ

Problem O. Shooting game (Division 2 Only!)

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

The simplest shooting game is a serie of battles between two players, and the only allowed weapons are small missiles.

Each player controls a tank, and the objective is to destroy the opponents tank by shooting it. The players take turns in shooting, and control the angle and initial velocity of their missiles. The initial velocity can never exceed 300.0 m/s, and can of course never be negative.

In order for a projectile to hit, it must have the correct velocity and angle to hit the opponent. The gravity acceleration is always $-9.8m/s^2$ along the y-axis, and there may also be wind. To keep things simple, we assume that the wind gives the projectile a constant acceleration along x-axis.

At current level, your weapon is damaged, so shooting angle is fixed. All that you can to do — set the velocity of the shot.

Input

First line of input file contains one integer T ($1 \leq T \leq 1000$) — number of test cases. Each test case is described by a line with 6 numbers x_u, y_u, x_o, y_o, w, d (each of them have type `double`). Your tank is positioned at (x_u, y_u) in meters, and your opponents at (x_o, y_o), where $0.0 \leq x_u < x_o \leq 1000.0$ and $0.0 \leq y_u, y_o \leq 800.0$. The number $-2.0 \leq w \leq 2.0$ gives the acceleration in m/s^2 of the projectile along the x-axis caused by the wind.

The angle of your weapon is given by $0 < d < 78$ in degrees. An angle $d = 0$ implies a shot along the increasing x-axis, and $d = 90$ would have implied a shot straight up.

Output

For each test case output in a new line initial velocity within the bounds which will ensure a hit, with precision 10^{-5}. If this is not possible, output -1.

Example

Standard input
```
2
24.980037311710223 288.98374755660933 382.68264850317206 369.12118991314696 0.0 21.444206874614686
390.2686697386107 201.33079962137526 440.4560006234233 692.7031898095327 0.0 51.0674695364569
```

Standard output
```
109.49391
-1
```