Problem A. Black-white balls

Input file:	`black-white-balls.in`
Output file:	`black-white-balls.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

n black and white balls were put into a bag. Petya doesn't know exactly how many black balls are there among them. He knows, however, that there are $0, 1, \ldots, n$ black balls among all balls in the bag with equal probability.

Petya took l balls from the bag at random, and l_1 of them turned out black, while l_2 other turned out white ($l_1 + l_2 = l$). Now he wants to predict how many black balls were there initially in the bag. Of course, if $l < n$, he can't be sure in his prediction, but he wants to predict a segment $[a, b]$, such that the amount k of black balls belongs to it with probability at least p.

You are given n, l_1, l_2 and p, and you must find such a and b that $b - a$ is minimal possible. If there are several such pairs (a, b), choose the one with the smallest a.

Input

In the first line there are four integer numbers: $1 \le n \le 50$ — the number of balls in the bag, $0 \le l_1 \le n$ — the number of black balls out of the l balls that Petya took from the bag, $0 \le l_2 \le n - l_1$ — the number of white balls that Petya took from the bag, $0 \le p \le 100$ — the required confidence in percent.

Output

Output numbers a and b separated by a space, $0 \le a \le b \le n$.

Sample input and output

black-white-balls.in	black-white-balls.out
50 1 24 100	1 26
50 1 49 100	1 1
50 1 10 95	1 15

Problem B. Chameleons All Around

Input file:	`chameleons.in`
Output file:	`chameleons.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

There are n chameleons moving along a circle of length L with speed 1, each moving either clockwise or counter-clockwise. i-th chameleon is initially located at point p_i and has color c_i, which is represented by an integer number.

When two chameleons meet, the following happens:

1. The one that was traveling counter-clockwise changes color to the color of the clockwise one.
2. They turn around, so the one that was traveling counter-clockwise now travels clockwise, and vice versa.

It's not so difficult to figure out that the ordering of chameleons among the circle always stays the same, and it is impossible for more than two chameleons to meet at the same point.

What will be the locations and colors of all chameleons at time T (the input data describes time 0)?

Input

The first line of the input file contains an integer n, $1 \le n \le 100\,000$. The second line contains an integer L, $1 \le L \le 10^9$. The next n lines contain 3 integers each, p_i, c_i, d_i — the location, color and direction of i-th chameleon, $0 \le p_i < L, 1 \le c_i \le 10^9, d_i$ is either -1 or 1. The last line contains an integer T, $0 \le T \le 10^{18}$.

The coordinate system is chosen in such a way that increasing one's coordinate means moving clockwise, and points on the circle have coordinates between 0 (inclusive) and L (exclusive). d_i can take one of two values: 1 for moving clockwise, and -1 for moving counter-clockwise.

All p_i are different.

Output

Output n lines with 3 numbers each, one floating-point (location) and two integers (color and direction), having the same meaning as the input numbers, but describing time T. The chameleons should be described in the same order as in the input file. It is guaranteed that no chameleons will meet at exactly T. Your answer will be accepted when each number is within 10^{-9} relative or absolute error of the correct answer. Please note that the coordinate you output should always be more than or equal to 0 and strictly less than L. Your answer will not be accepted when you output a coordinate very close to L but the correct answer is 0.

Sample input and output

chameleons.in	chameleons.out
4 13 2 1 1 0 2 -1 12 3 1 5 2 1 23	2.000 2 1 12.000 1 1 9.000 3 1 3.000 3 -1

Problem C. Distinct Substrings

Input file:	`distinct.in`
Output file:	`distinct.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

A well-known application of suffix trees is solving the following problem: given a string, find the number of distinct substrings that the string has. For example, the string "abac" has 9 distinct substrings: "a", "b", "c", "ab", "ba", "ac", "aba", "bac", "abac".

You are faced with generating testcases for this problem.

More specifically, you should find the shortest string consisting only of lowercase English letters that has exactly the given amount n of distinct substrings. Among several shortest strings, choose the lexicographically smallest one.

Input

First and only line of the input file contains an integer n, $1 \le n \le 300$.

Output

In the only line of the output file write the sought string.

Sample input and output

distinct.in	distinct.out
5	aab

Problem D. Fermat's Last Theorem

Input file:	`fermat.in`
Output file:	`fermat.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

Given a positive integer n and a positive prime number p, find x, y and z such that $x^n + y^n = z^n$ modulo p and x, y and z are nonzero modulo p or report that there's no such triple.

Input

The first line of the input file contains the number t of testcases to solve, $1 \le t \le 1000$. Each of the next t lines contains two integers n and p, $3 \le n \le 10^6, 2 \le p \le 10^6$.

Output

For each input testcase, output one line:

- when there exists a solution, output three integers x, y and z, $1 \le x, y, z \le p - 1$. If there are multiple solutions, output any.
- when there's no solution, output one integer -1.

Sample input and output

fermat.in	fermat.out
2 5 41 3 5	-1 1 2 4

Problem E. Friendly Points

Input file:	`friends.in`
Output file:	`friends.out`
Time limit:	10 seconds
Memory limit:	256 megabytes

Consider n distinct points on a plane.

Two points from that set are said to be *friends*, when there exists a rectangle with sides parallel to coordinate axes that contains those two points and doesn't contain any other point from the given set. A rectangle is said to *contain* a point if the point lies within the rectangle or on its border.

How many pairs of friends are there among the given points?

Input

The first line of the input file contains an integer n, $1 \le n \le 100\,000$.

The next n lines contain two integers each, the coordinates of the given points. The coordinates don't exceed 10^9 by absolute value.

Output

Output one integer number — the sought number of pairs of friends.

Sample input and output

friends.in	friends.out
5 0 0 0 2 2 0 2 2 1 1	8

Problem F. Maximal Clique

Input file:	`maxclique.in`
Output file:	`maxclique.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

This is the moment you've been waiting for all your life: you've invented a way to quickly solve the Maximal Clique problem: given an undirected graph find the size of the maximal subset of its vertices that form a clique (are pairwise connected). This problem is NP-hard, meaning you've got a proof that P=NP!

Unfortunately, the scientific community is not so eager to listen to you. Your papers on the subject are being rejected because of "solving an obviously unsolvable problem". Your phone number is already on the ignore list of all Computer Science professors you know. The world seems to hate you.

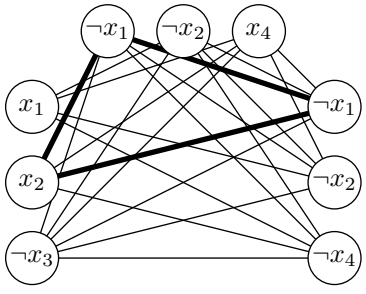
So you've decided to create a solver for the Maximal Clique problem and put it online, so that everyone can check for himself that you're right. You've already implemented the solver and almost launched the website, but then you've realized that this is not a very good idea: if you make the solver available, everyone will be able to solve every problem from NP by reducing it to the Maximal Clique problem. What if people will just silently use it instead of bringing you fame and respect?

Luckily, the only proof of NP-hardness of the Maximal Clique problem you know works by reducing the 3-SAT problem to it in a very specific way. So you've decided to check if the input graph given to your solver could be obtained from this reduction, and if yes, refuse to solve the problem. That way, nobody will be able to get quick solutions for all problems from NP, but everyone will still be able to verify your solver by feeding other graphs to it.

3-SAT problem statement is: given a formula of form $(t_1^1 \vee t_2^1 \vee t_3^1) \wedge (t_1^2 \vee t_2^2 \vee t_3^2) \wedge ... \wedge (t_1^n \vee t_2^n \vee t_3^n)$, where each term t_j^i is either some boolean variable or its negation (more formally, either x_k or $\neg x_k$), check whether there exists some assignment of true/false values to each variable so that the formula evaluates to true. All three terms in one clause must represent different variables.

The reduction works in the following manner. From the above formula, we create a graph with $3n$ vertices, one for each variable of each clause. Two vertices corresponding to terms t_j^i and t_s^r are connected when $i \neq r$ (so the terms belong to different clauses) and those terms are non-contradictory (they are either equal or represent different variables).

The following picture shows the resulting graph for the formula $(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4)$:



Now a clique of size n corresponds to a valid true/false assignment that satisfies at least one term in each clause. The edges highlighted on the above picture form a clique of size 3 and show that setting x_1 to false and x_2 to true satisfies all clauses, irrespective of the values of x_3 and x_4.

Given a graph, you need to check if it could be created by the above reduction. The vertices are permuted arbitrarily.

Input

The first line of the input file contains two integers v and e, $1 \leq v \leq 100$, denoting the number of vertices and edges in the graph. The next e lines contain two integers each, denoting the numbers of vertices connected by an edge. Each pair of vertices are connected at most once, no edge connects a vertex to itself.

Output

Output "YES" when the given graph could be obtained by the given reduction, or "NO" otherwise.

Sample input and output

maxclique.in	maxclique.out
9 22 1 3 1 6 7 1 8 9 9 1 2 3 2 4 2 5 2 6 2 8 3 4 3 5 3 7 4 8 4 9 5 6 5 7 5 8 5 9 6 7 6 9 7 8	YES

Problem G. Polygon

Input file:	`polygon.in`
Output file:	`polygon.out`
Time limit:	5 seconds
Memory limit:	256 megabytes

You are given lengths of sides of some polygon. You must find the infimum of the possible areas of simple polygons with such side lengths. *Infimum* of a set of real numbers A is the exact upper bound of the set L of all real numbers y such that for any $x \in A$ holds $y \le x$.

A *simple polygon* is a polygon without self-intersections and self-touchings.

Input

The first line contains integer n, $3 \le n \le 10$ — the number of sides of the polygon. The second line contains n integers $a_1, a_2, \ldots, a_n$, such that $2a_i < \sum_{j=1}^{n} a_j$ for any $1 \le i \le n$ (this means that there exists a simple polygon with sides $a_1, a_2, \ldots, a_n$. Also, $1 \le a_i \le 100$.

Output

Output one real number — the answer to the problem. Your answer will be considered correct if absolute or relative error is less than 10^{-6}.

Sample input and output

polygon.in	polygon.out
3 3 4 5	6.0000000000
4 8 4 3 5	4.4721359550
10 5 5 5 5 5 5 5 5 5 5	0.0000000000

Problem H. Recover path

Input file:	`recover.in`
Output file:	`recover.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Traveller Gregory is famous for his ability to always choose the shortest path for his journey. Ostap is a journalist who seeks for information about the recent Gregory's trip. He managed to get the evidence that during this trip Gregory visited a number of cities. However, there is no information about the order in which they were visited, and no information about the starting city and the ending city of Gregory's trip (the entire trip is (one of) the shortest path between these cities). Help Ostap to find any shortest path that contains all specified cities.

Country in which Gregory traveled consists of n cities and m undirected roads between them. For each road Ostap knows the time it takes to travel it, and the "shortest" word above is with respect to those times.

It is guaranteed that there exists some shortest path going through all specified cities.

Input

First line contains two integers n, m ($1 \le n, m \le 10^5$). Each of the m following lines contains a description of a single road a_i, b_i, t_i ($a_i \ne b_i$, $1 \le a_i, b_i \le n$, $1 \le t_i \le 10^4$) means Gregory can go between a_i and b_i by road and that will take t_i seconds. The next line contains k — the number of cities that Ostap knows Gregory has visited. The last line contains a list of these cities. All cities in that list are distinct.

Output

On the first line output the number of roads in the sought shortest path. On the second line output the list of road numbers (numbered in the order of appearing in the input) in the order of that shortest path. If there are many solutions, output any.

Sample input and output

recover.in	recover.out
6 6 1 2 2 2 6 2 1 3 1 3 4 1 4 5 1 5 6 1 3 5 1 3	3 3 4 5
6 6 1 2 2 2 6 2 1 3 1 3 4 1 4 5 1 5 6 1 2 1 6	2 1 2

Problem I. Schedule

Input file:	`schedule.in`
Output file:	`schedule.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

From the very early days of the Kingdom one of the most honorable duties for its citizen was the duty of the Royal Guardian. The Guardians serve as personal bodyguards for the King. At any moment of time exactly one of the Guardians was assigned to perform this duty.

Recently the King have issued the new labour rules. According to these rules the five day working week (from Monday to Friday) became mandatory for all citizens. Each working day starts at 9:00 in the morning and lasts until 18:00 in the evening. Any duties in non-working hours are subject for additional reward, however it is not allowed to work more than T non-working hours during one calendar week. A week starts on 00:00 on Monday and ends on 24:00 on Sunday.

The Labour Union of Guardians has decided to check the schedule of duties of the Royal Guardians, and you were chosen to help them.

You are given the initial schedule of duties, and a list of updates to this schedule. Find the fraction of time during which the entire schedule was valid according to the labour rules. You are only interested in time interval from T_1 to T_2, so you ignore all the duties outside this interval.

Input

The first line of the input file contains three integers N, M and T. Here N ($1 \le N \le 10^5$) is the number of entries in the schedule, M ($0 \le M \le 10^5$) is the number of updates to the schedule, and T ($0 \le T \le 123$) is the number of hours that one is allowed to work during the non-working hours per week.

The second line contains two dates — T_1 and T_2 (It is guaranteed that T_1 comes earlier than T_2).

The next N lines describes the initial schedule of the Guardians. Each line consists of a date D_i and the name of the Guardian that starts his duty at time D_i ($1 \le i \le N$, it is guaranteed that D_is are in ascending order, and D_1 comes no later than T_1). (The name is a case-sensitive sequence of English characters.) After starting his duty at time D_i the Guardian serves until the next Guardian comes to replace him.

Each of the following M lines describes the schedule of updates. Each update is described by A_i — the date when this update become active, B_i and E_i (the time slot of the update), and the Guarding name (in the same format as above). When this update becomes active, the schedule is modified in the following manner: this guardian will now serve the duties between B_i and E_i, the schedule outside this time slot is not affected (see the note below the example for more explanations). It is guaranteed that A_i comes no later than B_i, and B_i earlier than E_i. All A_is are in non-descending order. When several updates happen at the same time, you should apply them in the order they're given in the input file.

All T_i, D_i, A_i, B_i and E_i in the input file are dates in format YYYY-MM-DD hh:mm. All dates are between 2009-01-01 00:00 and 2009-12-31 23:59, inclusive.

Output

Output just one number — the fraction of time from T_1 to T_2 during which the schedule was valid according to the labour rules. Your answer will be accepted when it's within 10^{-6} absolute or relative error of the correct one.

Sample input and output

schedule.in
2 1 2 2009-12-07 09:00 2009-12-07 22:00 2009-12-07 08:00 Vasya 2009-12-07 14:00 Vanya 2009-12-07 15:00 2009-12-07 16:00 2009-12-07 20:00 Vasya
schedule.out
0.53846153846153844

Note

Here's what happens in the example case. We're only interested in a part of a day, from 9 in the morning to 22 in the evening. Initially, Vasya is scheduled to perform the duties from 9 in the morning to 14 in the afternoon, and Vanya is scheduled to perform the duties from 14 in the afternoon to 22 in the evening. This means Vasya works only during the working hours, and Vanya gets all the hard work of 4 non-working hours, which is more than the maximum of 2 allowed. Thus this schedule is invalid according to the labour rules.

However, at 15 in the afternoon the schedule is updated. Now, Vasya is performing the duties from 9 to 14 and from 16 to 20, and Vanya is performing the duties from 14 to 16 and from 20 to 22. This means 2 non-working hours for each of the Guardians, which is allowed under the labour rules.

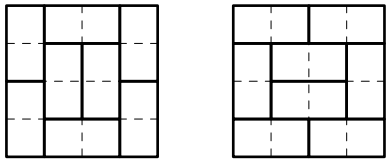
So the schedule was invalid from 9 in the morning to 15 in the afternoon, and valid from 15 in the afternoon to 22 in the evening, which corresponds to the fraction of 7/13.

Problem J. Cornerless Tiling

Input file:	`tiling.in`
Output file:	`tiling.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

An *cornerless tiling* of an $m \times n$ rectangle is such tiling of this rectangle with 1×2 and 2×1 dominoes that no four dominoes share a corner.

For example, here are the two possible cornerless tilings of 4×4 square:



How many cornerless tilings of $m \times n$ rectangle are there?

Input

First and only line of the input file contains two integers m and n, $1 \le n, m \le 1000$.

Output

In the only line of the output file write the sought number of tilings.

Sample input and output

tiling.in	tiling.out
4 4	2