Problem 1. Polesov and work

Input file:	input.txt or *standard input*
Output file:	output.txt or *standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

Viktor Mikhailovich Polesov, being a true representative of the intelligentsia, is intelligent and fast-thinking. He has recently learned that work of a force is the dot product of the force vector by displacement vector. Naturally, he wants to maximize the work, so that nothing goes to waste. All he has to do is choose where to move.

It is possible to move from the point $(0, 0)$ to any integer point in the circle $x^2 + y^2 \leq R^2$. The force vector is known — it is the same everywhere and has the coordinates (a, b). Find the maximum work that Polesov is so curious about.

Input

The first line of the input file contains a single integer T — the number of tests ($1 \leq T \leq 1\,000$). It is followed by T lines, each containing three integers a, b — force vector coordinates, and R — the radius of the circle ($-10^9 \leq a, b \leq 10^9$, $1 \leq R \leq 10^9$).

Output

For each test, print a single integer — the maximum work.

Example

input.txt or standard input	output.txt or standard output
3	20
10 -10 2	10
2 3 3	15
5 1 3	

Example explanation

In the first test, the circle with the center at the origin of coordinates and a radius of 2 contains 13 integer points. For the points $(2, 0)$, $(1, -1)$, $(0, -2)$, the dot product by the force vector $(10, -10)$ is maximal. For instance, $10 \times 1 + (-10) \times (-1) = 20$.

Problem 2. Orienteering

Input file:	`input.txt` or *standard input*
Output file:	`output.txt` or *standard output*
Time limit:	5 seconds
Memory limit:	256 mebibytes

Khodislav has taken up orienteering and is participating in a contest. The field is an endless plane without obstacles, and he can move at the same speed in all directions. There are N checkpoints in the field, which every participant must visit in the ascending order of their numbers. The check-in system is contactless — each checkpoint has a base station that automatically checks in any participant in a range less than or equal to R. It is guaranteed that checkpoint coverage areas do not overlap, but they can touch each other.

A participant starts at any point of the first checkpoint coverage and finishes at the moment of checking in at the last checkpoint. Participants are allowed to enter other checkpoint coverage areas on their way to the necessary checkpoint, but in this case, they are not checked in there.

Khodislav is feeling lucky and believes he will be able to cover the distance optimally. Help him calculate the distance he will have to cover.

Input

The first line of the input file contains two integers: N — the number of checkpoints ($2 \leq N \leq 100$) and R — the check-in radius ($1 \leq R \leq 10^9$).

Next come N lines describing the checkpoints in the required check-in order. Each line is a pair of integers x_i, y_i with the coordinates ($-10^9 \leq x_i, y_i \leq 10^9$).

Output

Print one real number — total distance passed if the route is optimal.

The relative or absolute error must not exceed **0.01**. This means that if the optimal answer equals X, your answer must differ from X by no more than $\frac{1}{100} \max(X, 1)$.

Examples

input.txt or standard input	output.txt or standard output
3 3 0 -3 0 100 0 50	141.0

Example explanation

Khodislav starts at $(0, 0)$, checks in at the second checkpoint at $(0, 97)$, turns and checks in at the third checkpoint at $(0, 53)$. The total distance covered is $97 + 44 = 141$.

Problem 3. Dice

Input file:	`input.txt` or *standard input*
Output file:	`output.txt` or *standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Khodislav is playing a tabletop role-playing game. He has finally chosen his weapons to deal with a monster and casts the crushing strike. To do this, he rolls dice, calculates the sum of numbers on their faces, and says it aloud to the game master.

Rolling a group of identical dice is characterized by three numbers n, f, and m, where n is the number of the dice, f is the number of faces on each die, and m is the modifier. The faces carry all numbers from 1 through f, and each and any face can be rolled; all rolls are independent. For instance, if $n = 3$, $f = 8$, $m = 5$, to define the sum, the player must roll three eight-faced dice, sum up the results, and add five: this is usually written as $3d8 + 5$.

The game master wants to check if Khodislav could get the sum he has reported after rolling the dice.

Input

The first line of the input file contains a single integer B — the number of strikes ($1 \le B \le 10^5$) cast by Khodislav. The following lines describe the strikes, one per line. First comes an integer S — the sum reported by Khodislav. It is followed by three integers: n, f and m describing the group of dice ($1 \le S \le 300$, $1 \le n \le 10$, $2 \le f \le 20$, $0 \le m \le 10$).

Output

For each strike in a separate line, in the same order as in the input file, print YES, if the sum was achievable, and NO otherwise.

Examples

input.txt or standard input	output.txt or standard output

Problem 5. Hockey

Input file:	`input.txt` or *standard input*
Output file:	`output.txt` or *standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Rules described in this problem differ from the conventional hockey rules.

A hockey match lasts 60 minutes, with two teams trying to score as many goals as possible. A hockey team consists of five field players and a goalkeeper.

Penalties are an important part of hockey. A field player can be given a penalty: in this case, the offending player leaves the ice for a period of time which depends on the violation. As the result, the number of players on the ice temporarily decreases for the team the offending player belongs to. There are two types of penalties in hockey: *major* and *minor*. A major penalty means the player leaves the ice for five minutes; with minor penalty, it is two minutes. When penalty time runs out, the player returns to the ice.

A minor penalty can be ended prematurely. A team is said to be playing short-handed when it has less players on the ice than the other team. If a team is playing short-handed and opponent scores a goal, then one of its players with minor penalty returns to the ice with his penalty expired ahead of time. If the team has several players with minor penalty, only the player who got the penalty first returns to the ice. If there are no players with minor penalties in the team, no one returns ahead of time.

Penalties during the game mean that the teams can play in various formats regarding the number of field players on the ice. We will denote the game format by `AxB`, meaning the first team currently has `A` field players on the ice, and the second team has `B`. For instance, in the beginning of the game each team has five players on the ice, and this format is denoted as `5x5`. If the first team currently has two players with penalty, and the second team has one, the format is denoted as `3x4`.

You are given a game protocol, registering the time of all penalties and goals. Calculate which formats happened during the game and for how long each format was played.

Input

The first line of the input file contains an integer N — the number of events in the match ($0 \leq N \leq 1\,000$).

The following N lines describe the events of the match, one per line. Events are described in the following format:

$$\texttt{mm:ss.d team type}$$

Where `mm:ss.d` — time of event with the precision of tenths of a second ($0 \leq \texttt{mm} \leq 59$, $0 \leq \texttt{ss} \leq 59$, $0 \leq \texttt{d} \leq 9$), `team` — team number (either 1 or 2), `type` — event type:

- `goal` — team scores a goal;
- `minor` — team player receives minor penalty;
- `major` — team player receives major penalty.

It is guaranteed that events of the type `goal` have non-zero decimal of a second, i.e. $\texttt{d} \neq 0$, and events of the type `minor` and `major` always have zero decimals of a second, i.e. $\texttt{d} = 0$.

Events are listed chronologically, i.e. they are arranged in the order of non-reduction of event times. It is guaranteed that at any moment of time each team has no more than 5 players.

Output

For each format of the game in which the teams have played non-zero time, print the format denotation and the time spent by the teams in this format in a separate line, separated by a space character. The format of time must be exactly the same as the format used in the input data. Lines can be printed in arbitrary order.

Example

input.txt or standard input	output.txt or standard output
10	4x3 00:47.9
06:41.0 1 minor	4x4 01:12.1
07:20.4 2 goal	4x5 06:39.4
22:22.0 2 minor	5x4 00:50.0
22:32.0 1 minor	5x5 50:30.6
23:00.1 1 goal	
23:12.0 2 minor	
23:59.9 1 goal	
41:02.0 1 major	
41:04.5 2 goal	
59:00.0 1 minor	

Example explanation

The game from the example had the following intervals:

- [00:00.0; 06:41.0) — until the first penalty, the game went in the initial format 5x5;
- [06:41.0; 07:20.4) — after a penalty, the teams were playing in the format 4x5 until the first team lost a goal while playing short-handed, and the player removed for a minor penalty returned to the ice;
- [07:20.4; 22:22.0) — the teams were playing in full numbers 5x5 until a penalty;
- [22:22.0; 22:32.0) — until the next penalty, the teams were playing in the format 5x4;
- [22:32.0; 23:00.1) — until a goal was scored, the teams were playing in the format 4x4, however, no players returned to the ice after the goal, because it was scored with equally-sized teams;
- [23:00.1; 23:12.0) — the teams continued playing in the format 4x4 until another penalty;
- [23:12.0; 23:59.9) — after that, the teams were playing in the format 4x3 until a goal was scored, and the second team player who had been penalized at 22:22.0 returned to the ice;
- [23:59.9; 24:32.0) — the teams were playing 4x4 until the first team player's penalty ran out;
- [24:32.0; 25:12.0) — the teams were playing in the 5x4 format until the second team player's penalty ran out;
- [25:12.0; 41:02.0) — the teams were playing in the full format 5x5 until a major penalty;
- [41:02.0; 41:04.5) — before the goal, the teams played in the format 4x5, but because a player of the team that lost a goal had a major penalty, that player does **not** leave the penalty box;
- [41:04.5; 46:02.0) — the teams continued playing as 4x5, until the first player's penalty ran out;
- [46:02.0; 59:00.0) — before the penalty, the teams were playing with all players 5x5;
- [59:00.0; 60:00.0) — teams ending the game in the format 4x5.

Problem 6. Days of the week

Input file:	`input.txt` or *standard input*
Output file:	`output.txt` or *standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

The Architect has created a multiverse with M universes. In each universe, the day of the week has been set individually. The Architect also has N buttons. Each button is connected to a certain set of universes. Pressing a button shifts all of its connected universes one day forward.

The Architect is curious: is there such a configuration of weekdays in the universes which is unachievable by any sequence of pressing buttons? Help the Architect solve this problem.

All universes use the Earth week: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday. When shifting the day of the week forward: Monday changes to Tuesday, Tuesday to Wednesday, etc, and Sunday changes to Monday.

The Architect can press buttons any number of times and in any order. When a button is pressed, all its connected universes change instantly and simultaneously.

Input

The first line contains the integer T — the number of tests in the input file ($1 \le T \le 500$). It is followed by the tests.

Each test begins with a line containting two integers: N and M — the number of buttons and the number of universes, respectively ($1 \le N, M \le 500$).

The second line of the test contains the initial configuration: which day of the week has been initially set in each universe. The universes are numbered from 1 through M: days of the week in the universes in the configuration are written in the same order.

This is followed by descriptions of buttons. For each button, a separate line defines which universes it is connected to, in the following format: the first number K defines how many universes are connected, and the following K numbers denote the numbers of these universes ($0 \le K \le M$). It is guaranteed that the specified universe numbers are different for each button.

It is guaranteed that the total number of buttons over all tests does not exceed 500, and the total number of universes over all tests also does not exceed 500.

Output

For each test, print an answer in a separate line.

If all 7^M possible configurations of days of the week are obtainable, print the word `NO` as the answer. Otherwise print `YES`, followed by any unachievable configuration, separated by a space symbol.

Examples

input.txt or standard input	output.txt or standard output

Problem 7. Balls

Input file:	input.txt or *standard input*
Output file:	output.txt or *standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

Alice and Bob are playing a game. They've got B blue and R red balls laid out in front of them. Alice has the first move, and after that, players alternate moves. Alice picks a single random ball and removes it. Bob removes a single red ball.

Alice chooses her balls randomly with equal probability regardless of their color. It does not matter which red ball Bob removes.

The game ends when one of the two outcomes occurs:

- there are no more blue balls — Alice wins;
- there are strictly more blue balls than there are red balls — Bob wins.

Alice and Bob would like a balance of outcomes, and are curious as for what number of blue balls is necessary for a game of $C = B + R$ balls for the probability of Alice winning h to be as close to 50% as possible. In other words, they want to minimize the value $|h - 0.5|$.

Input

The first line of the input file contains a single integer G — the number of games Alice and Bob are going to play ($1 \le G \le 10^5$).

The following lines define the number of balls C in each game ($2 \le C \le 2 \cdot 10^5$), one line per game.

Output

For each game in a separate line, in the same order as in the input file, print the number of blue balls necessary for the chance of Alice's victory to be as close as possible to 50%.

Examples

input.txt or standard input	output.txt or standard output

Problem 8. Romualdych and remainders

Input file:	`input.txt` or *standard input*
Output file:	`output.txt` or *standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Old man Romualdych learned about division with remainders and it just threw him off the hinges. The thing got him all mixed up and sweating like a pig. What a sad sight he was, hoping to find some number x in the interval $[a, b]$, that would produce the specified remainder r when divided by some number y. Let's face it — Romualdych is not the sharpest tool in the shed, and even if he sinks his few remaining teeth into the task, he is not likely to cope without your help.

Input

The first line contains a single integer T — the number of tests in the file ($1 \leq n \leq 200\,000$).

Each of the following T lines contains three integers: a, b — interval bounds, and r — required remainder ($0 \leq a \leq b \leq 10^{18}$, $0 \leq r \leq 10^{18}$).

Output

Print T answers in the same order as the tests in the input file are given, one answer per line.

Each answer consists of two integers x and y, such that $a \leq x \leq b$, $1 \leq y \leq 2 \cdot 10^{18}$, and the remainder from the division of x by y equals r. If there are several possible answers that fit all the requirements, choose any answer with the minimal x. If there are no possible answers, print two integers: $x = -1$ and $y = -1$.

Example

input.txt or standard input	output.txt or standard output
2	6 3
6 8 0	-1 -1
3 5 10	

Example explanation

In the first test, 6 is divided by 3, and the remainder is indeed 0. Since 6 is the smallest number in the interval $[6, 8]$, this is the correct answer. Instead, the following answer can be printed too: $x = 6$ and $y = 2$ (minimizing y is not required), while the answer $x = 8$ and $y = 4$ cannot be printed, because its x is not minimal.

In the second test, there are no answers, since it is impossible to get a remainder of 10 for x in the interval $[3,5]$ regardless of the y it is divided by.

Problem 9. Polynomials

Input file:	`input.txt or standard input`
Output file:	`output.txt or standard output`
Time limit:	3 seconds
Memory limit:	256 mebibytes

On the left side of the board, there are N polynomials, and on the right side, there are M polynomials. Your task is to *construct* each of the M polynomials from the right side of the board with the minimum number of actions.

To construct a polynomial from the right side, first choose any of the N polynomials from the left side. The following transformations can be applied to the chosen polynomial:

- Differentiation: a polynomial is replaced by its derivative.

- Integration: a polynomial is replaced by its antiderivative with an arbitrary integration constant.

Transformations can be applied in arbitrary order any number of times, however, one application counts as one action. You can use no transformations at all if a polynomial from the right side matches some polynomial from the left side.

Input

The first line of the input file contains two integers: N and M ($1 \leq N, M \leq 10^5$).

Each of the following $N + M$ lines describes the polynomials, one polynomial per line. The first N polynomials are polynomials from the left side of the board, the rest are from the right side of the board.

The description of the polynomial $a_0 + a_1 x + \cdots + a_K x^K$ begins with a nonnegative integer K — its degree. Next come integers $a_0, a_1, \ldots, a_K$, which are the coefficients of the polynomial ($-10^9 \leq a_i \leq 10^9$). Herewith, $a_K \neq 0$.

It is guaranteed that the sum of degrees of all polynomials on the left side of the board is not greater than 10^5. It is the same for the polynomials on the right side of the board.

Output

For each polynomial from the right side, print the minimum number of actions necessary for its construction. Print your answers one per line in the same order as the order in which the polynomials are listed in the input file.

Examples

input.txt or standard input	output.txt or standard output
2 1 2 1 1 1 2 7 6 2 2 7 6 1	4
2 1 2 1 1 1 0 1 1 1 1	1

Example explanation

In the second example there are two polynomials on the left side of the board: $p_1(x) = 1 + x + x^2$ and $p_2(x) = 7 + 6x + 2x^2$. We need to obtain the polynomial $q(x) = 7 + 6x + x^2$. In order to do that we apply differentiation to p_1 twice obtaining $p_1'(x) = 1 + 2x$ first, and $p_1''(x) = 2$ next. Now, let's integrate

$p_1''(x)$ with 6 as the integration constant producing $6 + 2x$. We integrate the result once again with 7 as the integration constant to get $7 + 6x + x^2$. We used 4 actions in total.

Problem 12. Nanoassembly

Input file:	`input.txt` or *standard input*
Output file:	`output.txt` or *standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

One of the problems of creating elements of nanostructures is the colossal time necessary for the construction of nano-parts from separate atoms. Transporting each of the vast number of atoms requires the use of a manipulator. While the manipulator is busy moving an atom, it can't move other atoms. Using a larger number of manipulators simultaneously is impossible, since manipulators are relatively large and don't fit into the space where the nano-parts are assembled. Experts at the Nanotech State University (NSU) lab proposed a method aimed at solving the problem described above.

The essence of the method consists in abandoning manipulators and substituting them with a semiplanar reflector developed in the laboratory. The device performs manipulations with atoms located on one of the sides of a graphene sheet, which can be viewed as a plane surface. The principle of the semiplanar reflector consists in folding the graphene sheet along a specified straight line, which brings all atoms located on a (specific) side relative to the line to the opposite side, occupying mirrored positions of their initial positions relative to the line. All atoms that were located on the other side of the folding line retain their initial positions. Once all atoms have been transferred, the graphene sheet is unfolded. As the result of several subsequent folding operations, a large number of atoms can be transferred.

Define the positions of the atoms after the manipulations based on their initial position on the graphene sheet surface and the description of the sequence of operations performed by the semiplanar reflector.

Input

The first line of the input file contains two integers N and M — the number of atoms and the number of operations ($1 \le N \le 10^5$, $1 \le M \le 10$).

The following N lines define the initial positions of atoms. Each position is described by two space-separated integer coordinates x and y, which do not exceed 10^4 in absolute value.

The following M lines describe operations performed by the semiplanar reflector. Four space-separated integers x_1, y_1 and x_2, y_2 are used to describe each operation — the coordinates of the start and end points of the vector lying on the folding line (the absolute value of each of the coordinates also does not exceed 10^4). The start and end points are different. The operation transfers atoms located in the semiplane to the right of the defined vector.
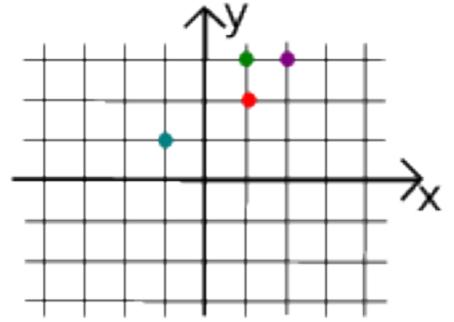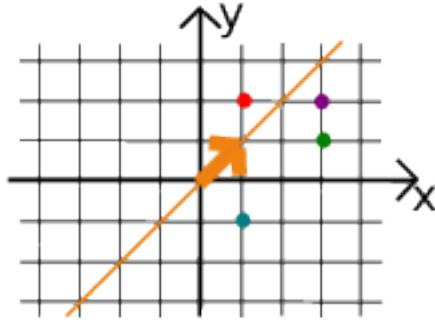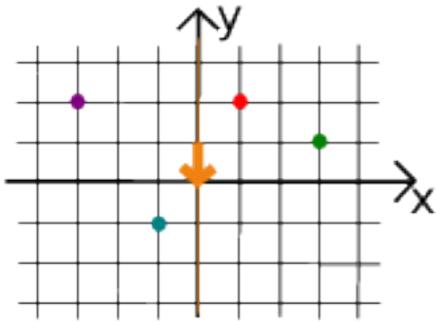
Output

The output file must contain N lines containing the coordinates of the atoms in the same order as in the input data, with absolute error not exceeding 10^{-5}.

Examples

input.txt or standard input	output.txt or standard output
4 2	1.0000000000000 2.0000000000000
1 2	1.0000000000000 3.0000000000000
3 1	-1.0000000000000 1.0000000000000
-1 -1	2.0000000000000 3.0000000000000
-3 2	
0 1 0 0	
0 0 1 1	

Note

Problem 13. Playoff

Input file:	`input.txt or standard input`
Output file:	`output.txt or standard output`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Let's consider a tournament based on the Olympic system. There are 2^N participants. They are split into pairs. The winner of each pair continues to the next round. The number of participants of that round is 2^{N-1}. They are split into pairs again, the winner goes to the next round and so on, until the absolute winner is established.

Let's introduce *stronger*, a relationship of teams. If the team A wins in the game of A versus B, then the team A is *stronger* than the team B. Let's consider *stronger* a transitive relationship: if the team A is *stronger* than the team B, and the team B is *stronger* than the team C, then the team A is *stronger* than the team C.

For instance, if the team A beats the team B in semifinals, and C beats D, and then A beats C, we can tell that A is *stronger* than D, however, we can't tell which of the teams B and C is *stronger*.

You are given the results of the tournament based on the Olympic system. Also given are several pairs of teams; define whether one team from each pair is *stronger* than the other.

Input

The first line of the input file contains an integer N ($1 \leq N \leq 18$).

The following 2^N lines contain the names of the participating teams. The names are unique and consist of no more than 20 small latin letters.

In the first round, in the first pair the first team plays with the second team, in the second pair "— the third and fourth and so on. In the second round, the winner of the first pair plays with the winner of the second pair, the winner of the third pair plays with the winner of the fourth pair and so on.

Pairs for the next rounds are matched in the same manner.

All games in the tournament are numbered according to the numeration of pairs in rounds: the numbers from 1 to 2^{N-1} are assigned in order to the pairs from the first round, the numbers of $2^{N-1} + 1$ to $2^{N-1} + 2^{N-2}$ are assigned in order to the pairs of the second round, etc.

The following line consists of $2^N - 1$ symbols 'W' or 'L' and describes the results of games in the given order, where the symbol 'W' denotes the victory of the first team in a pair, and the symbol 'L' "— the victory of the second team.

The next line contains an integer Q "— the number of queries ($1 \leq Q \leq 10^5$).

The following Q lines contain query descriptions. Each query consists of two different space-separated words "— the names of the teams.

It is guaranteed that the size of the input file is equal to or less than three megabytes.

Output

The output file must contain one of the following words for each query:

- `Win`, if the first team *stronger* than the second team;

- `Lose`, if the second team is *stronger* than the first team;

- `Unknown`, if it is impossible to tell that one of the teams is *stronger* than the other.

Examples

input.txt or standard input	output.txt or standard output
3	Unknown
cska	Win
jokerit	Lose
ska	
dynamo	
avangard	
akbars	
sibir	
metallurg	
WWLWLWW	
3	
jokerit avangard	
ska metallurg	
metallurg akbars	

Note

Tournament rounds in the example looks like this:
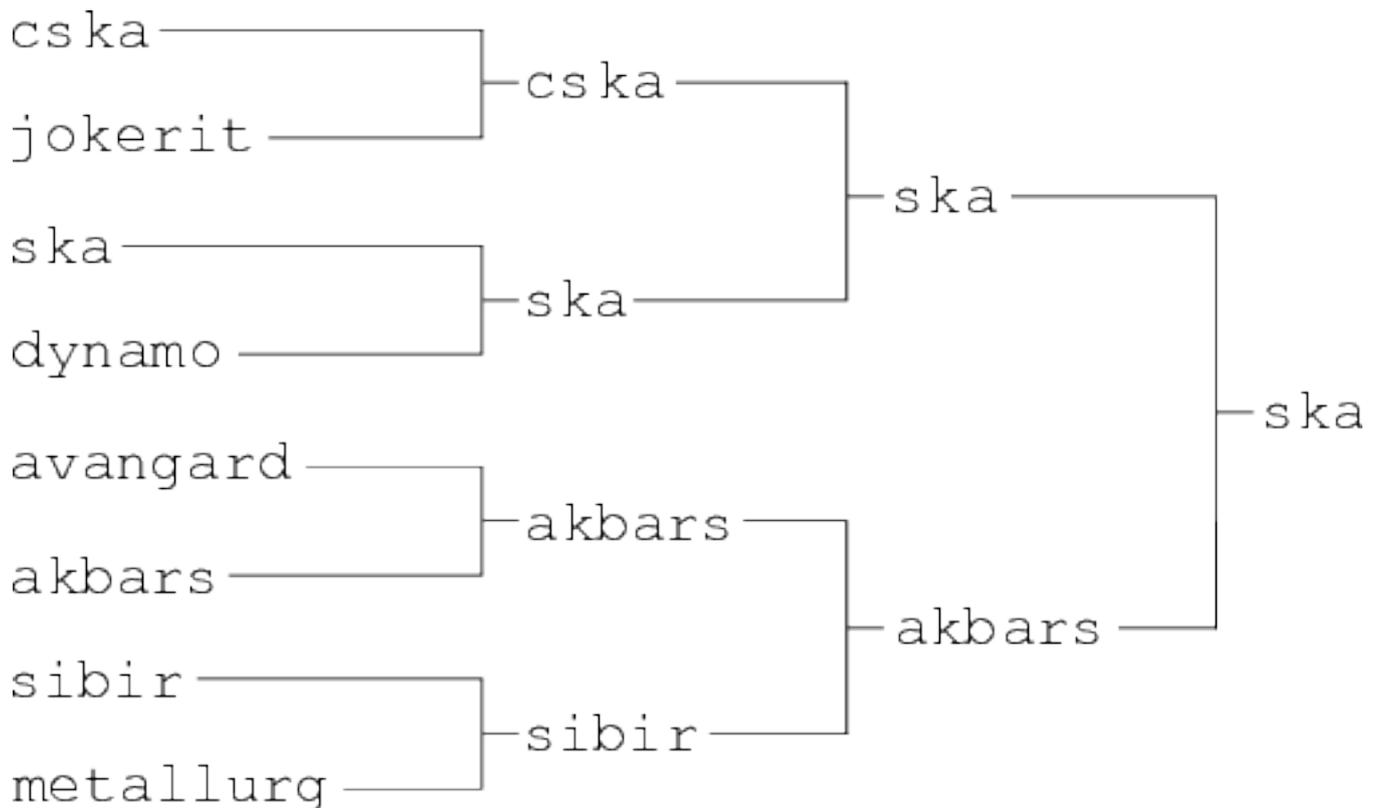
Problem 14. Master Gambs' Chairs

Input file:	`input.txt` or *standard input*
Output file:	`output.txt` or *standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Father Theodor once visited a furniture show, where they sell furniture of all sorts, made by master Gambs. Needless to say, he is very interested in a particular set of chairs from the house of Vorobyaninov, the head of the nobles. On the one hand, he knows from the head that all furniture at his home was marked, — ¡¡in case of all them guests¿¿, — on the other hand, he's got no idea what the particular set looks like. Here's a plan he's cooked up: he'll buy chairs from those sets (at least one chair from each set), something he's got the money to buy, and check them. If he finds a mark, then he can buy the remaining chairs from the desired set later on the new money he would get from his ¡¡mother¿¿.

Naturally, he'd like to check as many different sets as possible, but he's short on money, and he's slightly out of his mind, having spent so much time lurking in the mountains. The task can prove too difficult for him. Help him out! Determine how many sets can he check, so that he bought at least one chair from each set, and he had enough money on all the purchased chairs.

Input

The first line of the input file contains two integers N and S, where N — the number of chairs on sale, S — the available sum of money ($1 \le N \le 10^5$, $1 \le S \le 10^6$).

The following N lines each contain two integers a and b — the index of the set the chair belongs to, and its price, respectively ($1 \le a \le N$, $1 \le b \le 10^5$).

Output

The output file must contain a single integer — the maximum possible number of furniture sets from which it's possible to buy pieces with the total cost not greater than S.

Examples

input.txt or standard input	output.txt or standard output
6 17 2 5 1 5 2 6 2 3 4 400 5 230	2

Note

Father Theodor can buy one piece from the first set, one or two pieces from the second set. He has not enough money to buy anything from the other sets.