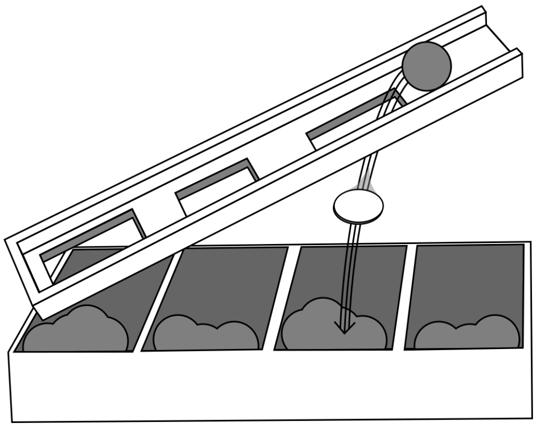
Problem A. Automatic Banking

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

The Bank of Byteland has developed an advanced technological solution to the problems it has with counting money deposited by clients.

The monetary system in Byteland is complicated: there are s possible nominals of coins (between 1 and s bytalers, respectively), but for each nominal may exist more than one type of coin with this nominal (for example, 1-bytaler coin depending of year may have thickness $2mm$ and mass 3 grams, or thickness $1mm$ and mass 4 grams).

To sort the coins, the machine works by running each individual coin along a sloped track. At every integer multiple of centimetres along, starting from 1cm, there is a slot in the track with a box underneath.



The slot will allow a coin to fall in, if the thickness of the coin (in millimetres) is **less than or equal to** the width of the slot (also in millimetres), and the mass of the coin (in grams) is **greater than or equal to** the trigger mass of the slot (also in grams). Its allows the coins to be sorted, because **1-based index of box the coins falls in is equal to its nominal in bytalers**.

You are given a list of the coins that will be deposited. Check the amount of deposited money in bytalers.

Input

First line of the input contains one integer — the number of slots s ($1 \leq s \leq 10^5$). Then s lines follow, the ith of those lines describes the slot for i-bytaler coins and contains the width of a i-th slot in millimetres and trigger mass in grams for the ith slot, a_i and b_i respectively ($1 \leq a, b \leq 10^5$).

Next line contains one integer c ($1 \leq c \leq 10^5$), the number of coins deposited. i-th of following c lines contains the thickness in millimetres and mass in grams of the jth coin, u_i and v_i respectively ($1 \leq u, v \leq 10^5$).

It is guaranteed that every coin will be able to fall into at least one box.

Output

Print one integer — amount of deposited money in bytalers.

Examples

standard input	standard output

1
4 4
1
2 6
	1

3
2 3
1 4
1 1
2
2 3
1 1
	4

3
3 3
2 4
1 1
2
3 3
2 3
	2

5
26 31
23 22
44 24
45 53
103 10
5
11 54
23 15
40 26
52 39
50 56
	19

Problem B. Basketball Competitions

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

The teams in Byteland high school basketball league aren't particularly good, but they make up for it in enthusiasm. The league is going to organise a single-elimination knockout tournament between them, where the 2^n teams play n rounds. In each round, the $2i+1$th remaining team pairs up with the $2i+2$th team and one or the other team is eliminated.

Each team has a scalar skill level. In the normal course of things, a team with higher skill level will always beat a team with lower skill level. However, training plays a part too: if one team studies another, learns its techniques, and trains against them, it can win.

The number of hours a team with skill a must train to beat a team with skill b (where $a \leq b$) is $|b-a|^2$. This training only affects that one game (it does not transfer to other teams).

You would quite like team from your school to win the tournament. If you take complete control over how every team trains, you can always make this happen. What is the minimum number of hours needed, in total across all teams, in order for your team (team 1) to win?

Input

First line of the input contains the integer r ($1 \leq r \leq 14$), the number of rounds in the tournament. Second line contains 2^r integers $s_1 \ldots s_{2^\mathrm{s}}$ ($0 \leq s_i \leq 10^6$ for each i), where s_i is the skill level of the ith team.

Output

Print the smallest number of hours needed for team 1 to win the tournament.

Examples

standard input	standard output
1 20 20	0
3 3 1 4 1 5 9 2 6	19

Problem C. Coins Distribution

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

You want to sell n ancient coins from your collection. To do that, you are planning to sell sets of exactly k coins. Because numismatists will not buy the sets with two similar coins, you plan to make all the coins in one set distinct.

You know types of each coin; two coins of one type are considered to be similar, two coins of distinct types — distinct. Your task is to prepare maximum number of k-coin sets, containing pairwise distinct coins.

Input

First line of the input contains two integers n and h ($1 \le h \le n \le 10^6$), the number of coins for sale and the number of coins in one lot.

Second line contains n integers $v_1, \ldots, v_n$ ($1 \le v_i \le 10^6$) — the types of the coins in no particular order.

Output

If it is not possible to make any k-sets at all without two coins of same type, output `impossible`.

Otherwise, output k lines (where k is the maximum possible number of sets) each containing h numbers from the input. You may not use any number any more times than it appears in v.

If there are multiple answers with a maximum value of k, you may output any one of them.

Examples

standard input	standard output
6 3 1 9 1 9 22 33	1 9 22 1 9 33

12 3
3 1 4 1 5 9 2 6 5 3 5 8
	1 3 5
	1 4 6
	2 5 8
	3 5 9

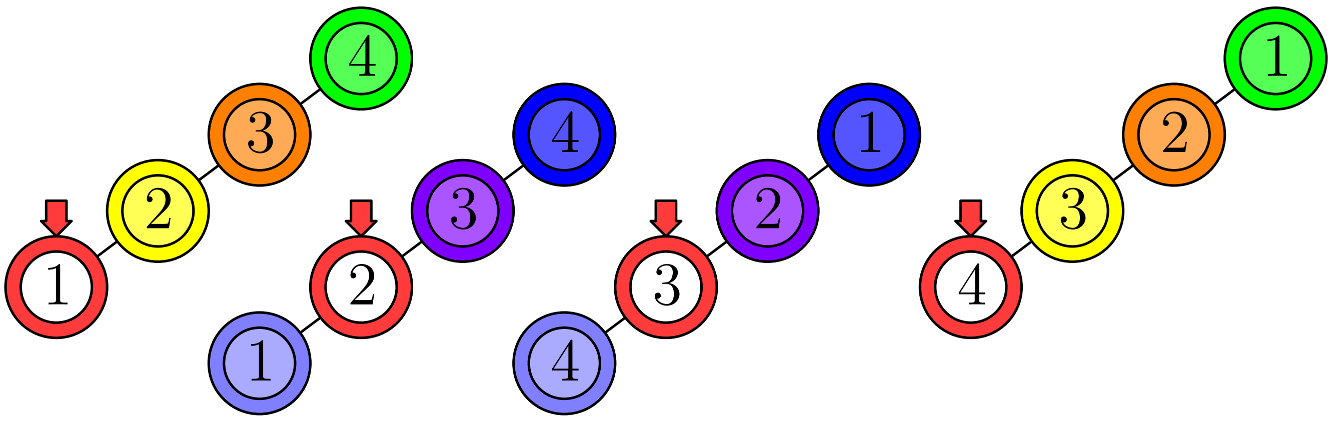
6 4 1 2 4 1 4 2	impossible

Problem D. Drawing of Electropolitan

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

The Bytesburg just finished construction of its new transport network, Electropolitan. There are n stations and exactly one way to get between any given pair of them; this is because there are only $n-1$ direct connections between pairs of stations.

You have been hired to put together the signage for each of the stations which shows where on the network a passenger is with a big arrow pointing to the bright red station in the centre.



Because the drawings of the Electropolitan network are fairly crude, it is actually possible that you could use the same sign in more than one station, and just write a different permutation of labels for the station names.

If you want to make signage for the whole network, what is the minimum number of unique designs you will need?

Input

The first line of input contains the number of stations, n $(1 \le n \le 3 \times 10^5)$.

The following $n-1$ lines each contain two distinct vertex indices a and b $(1 \le a, b \le n)$ indicating that there is a direct route between these stations.

Output

Output the minimum number of map designs that can be made, such that for any station at least one of these map designs can be re-labelled such that this station is in the centre.

Examples

standard input	standard output
4	2
1 2	
3 2	
4 3	
11	10
1 2	
2 3	
3 4	
4 5	
4 6	
4 7	
5 10	
9 10	
8 10	
7 11	
7	7
1 7	
2 7	
2 3	
4 7	
4 5	
5 6	

Problem E. Economics of Football

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

Mino is one of most famous football agents in the world. He asks for 5% commission for every deal.

Mino organises a goalkeeper auction each year. Every team (numbered from 1 to n) must participate in this action, although making or an accepting an offer is optional. Everyone puts in bids for the goalkeepers they would like to hire, provided they can sell their current goalkeeper at the same time.

This is a very transparent process – Mino can see exactly how much commission he will make if he accepts the right buyers' offers on behalf of the sellers. He may discard some offers from buyers in order to drive up the overall commission. In fact, he might even decide to discard all of the offers from one team and let them stay with their current goalie, if it makes more money for him.

Find the maximum commission Mino can make if he discards offers optimally.

Input

The input consists of:

- one line containing two integers n and m ($1 \le n \le 150$, $0 \le m \le n \times (n-1)$), the number of teams and the number of offers made.
- m lines, describing the offers.

  The ith such line contains three integers f_i, h_i and a_i ($1 \le f_i, h_i \le n$, $f_i \ne h_i$, $0 \le a_i \le 10^6$), the team making the offer, the team that owns the goalkeeper that the offer is for and the amount offered.

  No team makes more than one offer to the same goalkeeper.

Output

Output how much Mino will earn via commissions if he discards offers optimally. Your answer must be accurate to an absolute or relative error of 10^{-6}.

Examples

standard input	standard output

`4 5`
`1 2 4`
`2 3 10`
`3 1 6`
`3 2 12`
`4 1 7`
	`1.10`

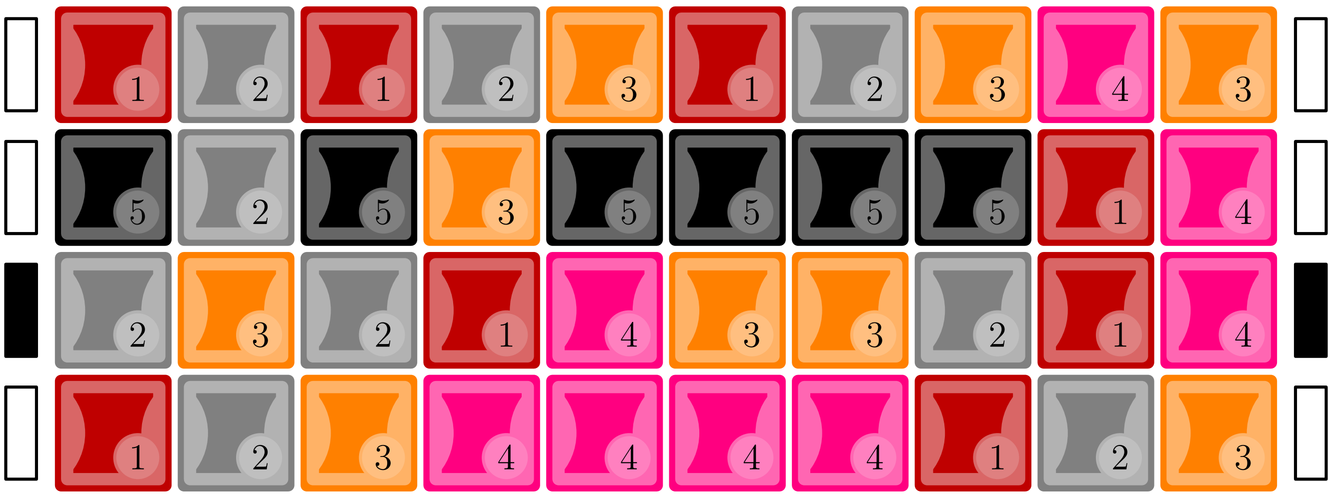
`4 5`
`1 2 14`
`2 3 8`
`3 1 4`
`3 2 10`
`4 1 5`
	`1.30`

Problem F. Flawed Grid

Input file:	*standard input*
Output file:	*standard output*
Time limit:	12 seconds
Memory limit:	512 mebibytes

Peter is the artist. The Peter's famous pictures made from square tiles arranged in a grid, at least for today's purposes. Peter likes to make pictures with exactly the same number of tiles of each colour.

Once Peter found at the cellar his first picture with flaws: it contained not the same amount of tiles of each colour. He decided to fix the art by removing some of the rows from it.



What is the greatest number of rows he can keep?

Input

The first line of input contains the number of rows, n ($1 \leq n \leq 40$), the number of columns, m ($1 \leq m \leq 10^5$), and the number of colours, c ($1 \leq c \leq 10^5$) in the art respectively.

Each of the next n lines contains m colours of cells $p_1 \ldots p_m$ ($1 \leq p \leq c$).

Output

Print the greatest number of rows that can be kept while keeping equal representation for each colour in the input, or 0 if no rows can be kept.

Examples

standard input	standard output

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

Problem G. Gregory's House

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

Gregory is popular doctor. He owns several ties, some of which he wear more than others. Because wearing all n at once would be impractical, he store the spares on a rack with $n - 1$ hooks. The first hook is a half metre from the door, the second one metre, the third a metre-and-a-half, and so on. This means that walking from and to the door to the ith hook involves i metres of walking.

Tonight Gregory plans to make a series of visits, each time wearing a tie appropriate to the proposed diagnosis. When Gregory comes back from one visit, he will take the tie he need from its hook and exchange it with the one he had been wearing before. This means that ties can move around as the night goes on.

Given the plan for tonight, and assuming Gregory is already wearing the first one, what is the best way to arrange the ties on the rack before setting off so as to minimise the number of metres walked?

Input

The first line of input contains two integers: c and n ($1 \le c, n \le 10^5$), the number of ties in total and the length of the itinerary respectively.

The second line of input contains n integers: $v_1 \ldots v_n$ ($1 \le v \le c; v[i] \ne v[i-1]$), the order of ties Gregory need to put on. Gregory is wearing the first one, and he never need to wear the same tie twice consecutively.

Output

Output the minimum number of metres Gregory need to walk, once he have optimised his tie rack.

Next, output $c - 1$ integers: an initial ordering of the tie rack that gives minimum walking distance, starting from place 1. This should include exactly once every tie except the first.

If there are multiple correct answers, you may output any one of them.

Examples

standard input	standard output
3 5 1 2 3 2 3	5 3 2
5 9 1 5 2 3 4 2 4 2 4	14 3 5 4 2

Problem H. Health Insurance

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Am insurance company invited two professors, A and B, for medical consultations. There are N patients in the department, all of whom have a cold and have to see both professors.

The patients are of three different types:

1. those who have to see professor A rst and B after that;
2. those who have to see professor B rst and A after that;
3. those who can see the two professors in either order.

The head of insurance company can estimate the duration of visit of each patient to each of the invited professors based on the medical history of the patient.

As the professors charge by time, the insurance company wants to minimize the total billable time. The professors bill for the time spent seeing the patients, but also for the time spent waiting for the next patient. The professors arrive together and also leave together. (The small print in the contract states that the professor who finishes first will also be paid for the time spent waiting for his colleague to finish.)

The task is to write a program to order the patients so as to minimize the total time billed by the professors.

Input

The first line of input contains the integer N ($1 \le N \le 20$). Each of the following N lines describes one patient as three integers: the type of patient (from 1 to 3), the duration of visit to professor A, and the duration of visit to professor B. The duration of each visit is a positive integer and does not exceed 1000.

Output

The first line of output should contain the total time billed by the professors. The second line should contain a permutation of the integers between 1 and N — the order in which professor A will see the patients. The third line should contain the order for professor B. If there are several optimal solutions, output any one of them.

Examples

standard input	standard output

3
3 5 7
1 6 1
2 2 6
	28
	1 2 3
	3 1 2

1 3 13 13	52 1 1

Problem I. Ink Jet

Input file:	`standard input`
Output file:	`standard output`
Time limit:	7 seconds
Memory limit:	512 mebibytes

You tried to refill the catridge of your Ink Jet printer, but somehow failed. Now spots of ink are beginning to form on the page and spread out.

The ink spreads out by forming infinitesimally-small blots on the page. A blot that appears at time t seconds after the incident grows in radius smoothly, at a rate of 1 cm per second, and may eventually overlap with other blots on the page.

At first the page is still usable, but when the combined size of the ink blots grows large enough, you will have to find another piece of paper.

How long will it take for this to happen?

Input

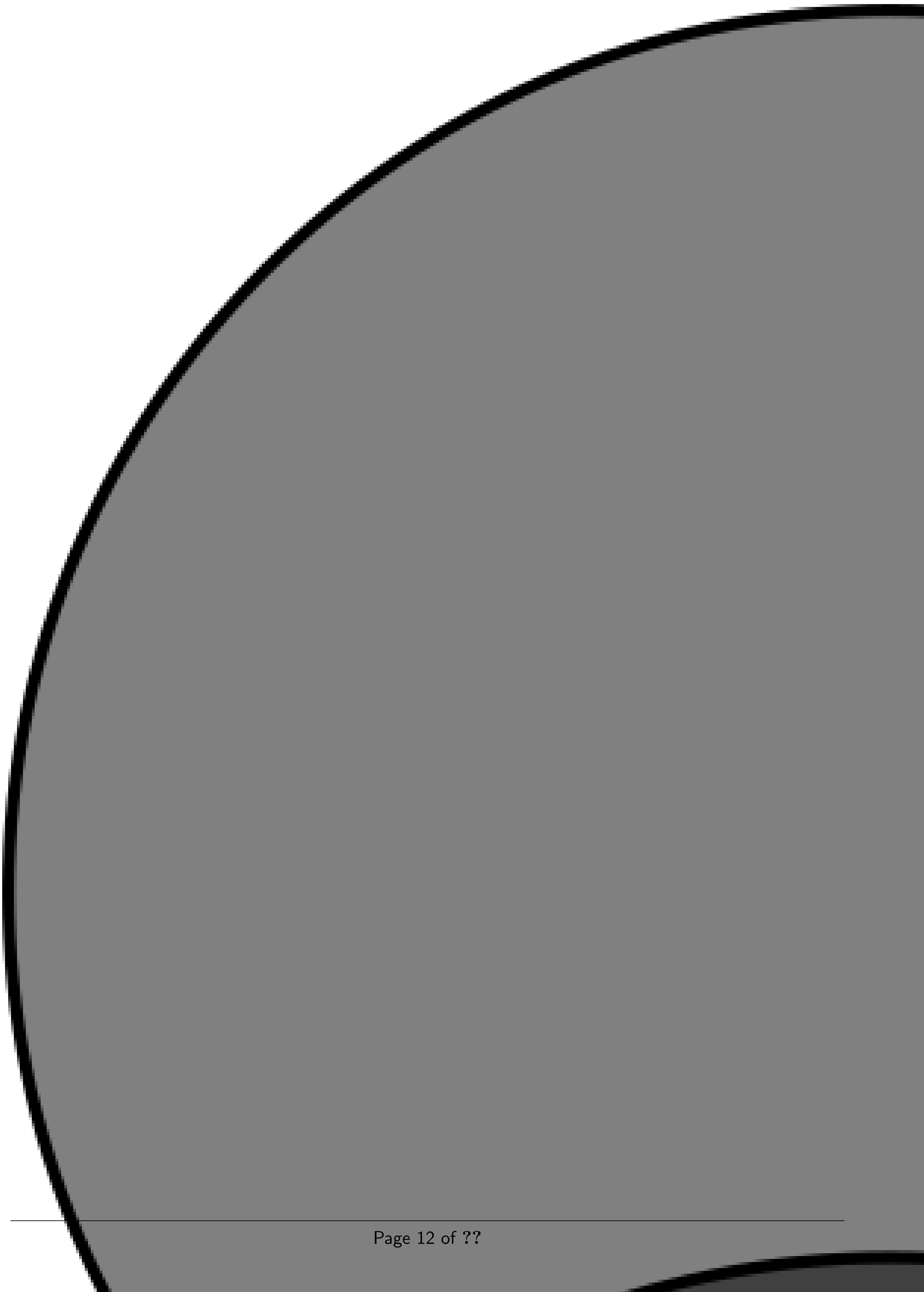
First line of the input contains an integer n ($1 \le n \le 100$), the number of inkblots, and the real-valued total area of ink in square centimetres at which the paper must be abandoned, a ($1 \le a \le 10^9$).

Each of n further lines, each with the x and y coordinates in centimetres of an ink blot ($-10^6 \le x, y \le 10^6$) and the time in seconds at which the blot first appears, t ($0 \le t \le 10^6$).

Output

Output the time in seconds at which the ink blots cover exactly a square centimetres of the infinitely-large page. Your answer must be accurate to an absolute or relative error of 10^{-6}.

Note



Examples

standard input	standard output
4 20.566371	1.414213614689
1.0 1.0 0.0	
1.0 3.0 0.0	
3.0 1.0 0.0	
3.0 3.0 0.0	
2 785.398163397	15.488715515659
-5 0 2	
5 0 3	
5 10000	52.552451206462
0 0 0	
0 0 1	
0 0 3	
12 0 1	
0 -6 2	

Problem J. Just Kidding

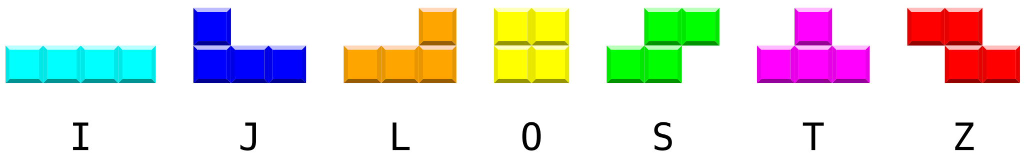
Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

This is an interactive problem.

Your task is to create the AI player for the so-called Brick Game, based on the well-known game *Tetris*.

Well, we are just kidding — your task is to complete at least one row.

Below you may see the set of rules for Brick Game:



- The playing field is a grid of width 10 and height 10, initially grid is empty.
- One by one, the game presents the player with a random sequence of tetromino shaped pieces, as seen in Figure **??**. The player may assume that **the first seven pieces are all distinct**, then the next seven pieces are **distinct**, and so on.
- The player may shift each piece left or right and rotate it by 90, 180 and 270 degrees. These movements take place *above* the grid.
- The piece then drops down into the playing area and locks into place as soon as some part of it cannot drop any further because part of some previous piece is in the way. It is *not* possible to shift or rotate the piece during the drop.
- The game ends as soon as one of the following happens:
  - **Some** horizontal row of the playfield is fully filled by the tiles, thus the player wins.
  - Some piece does not fully drop into the grid, the player loses.

Interaction Protocol

The judges program sends random piece (see the note below), denoted by one of the uppercase letters $\{\texttt{I}, \texttt{J}, \texttt{L}, \texttt{O}, \texttt{S}, \texttt{T}, \texttt{Z}\}$ as in Figure **??** above.

In reply to each piece your program must print two integers r and d, where r $(0 \le r \le 3)$ is rotation (0, 90, 180 and 270 degrees clockwise, respectively), and d is the **leftmost** column occupied by the piece (columns are enumerated starting at 1 from left to right).

Dont forget to print end-of-line character and to flush the output after each answer. For example, you can use `fflush(stdout)` in C++, `System.out.flush()` in Java, and `sys.stdout.flush()` in Python.

The game ends as soon as your program successfully completes a row or sends an bad placement, whichever happens first (if both happen at the same time, the bad placement takes precedence). A placement is bad if some part of the piece would have to be placed outside of the playfield, either to the top or to the side.

If your submission completes a row, interactor will send a single character `W`. Upon receiving this, your submission must terminate with exit code `0` as usual.

The seed values for this randomiser have been fixed in advance, such that every submission is run on the same piece sequences.

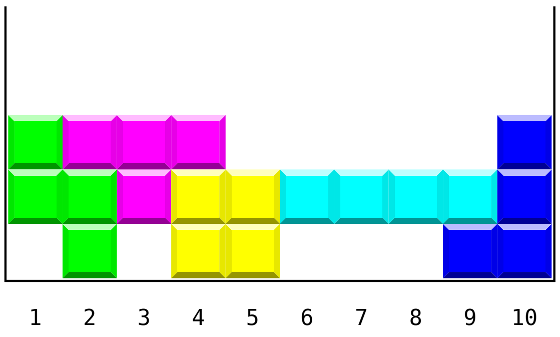
Example

standard input	standard output
S	
	1 1
O	
	2 4
T	
	2 2
J	
	3 9
I	
	0 6
W	

Note

In the example interaction above, possible output of the judge program is on the left and one possible correct output of a player is on the right.

The empty lines on both sides added only to emphasise the chronological order. The interactor will never output any empty lines.



Problem K. Knight or Liar?

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	512 mebibytes

There are n inhabitants on the Hope Island. Each of them is either a knight who always speaks the truth, or a liar whose statements are always false. Each of these people has a unique integer number from 1 to n assigned to him.

Once a year the people gather at the center of the island, and after that each of them makes exactly one statement of exactly one of the following forms: "inhabitant with a certain number is a knight" or "inhabitant with a certain number is a liar".

This year, Vasya came to visit the island at the start of the meeting. He heard and recorded some statements, after which the people went to take a lunch break. Right now, Vasya wants to simulate the further development of events.

Let a *protocol* be a list of n statements: the statement of the first inhabitant of the island, the statement of the second one, the statement of the third one, ..., the statement of the last inhabitant. In Vasya's model, each of the inhabitants which did not yet say anything will say one of the possible $2n$ statements independently and with uniform probability. The possible statements are: "first inhabitant is a knight", "first inhabitant is a liar", "second inhabitant is a knight", ..., "n-th inhabitant is a liar". After the protocol is complete, Vasya will search for a *solution*: for each of the inhabitants, he wants to determine whether this inhabitant is a knight or a liar, so that all statements of the protocol are correct.

Note that the number of possible solutions can be different for different protocols. In particular, there are protocols for which no solutions exist at all: for example, if a certain inhabitant says about himself that he is a liar, there are no solutions no matter what and about whom all the other inhabitants say. On the contrary, if in the protocol, each inhabitant says about himself that he is a knight, each of them can be either a knight or a liar independently of the others, so the number of possible solutions is 2^n.

Now Vasya asked you to estimate his chances to get a solution. Consider the moment of time when the protocol is already fixed, i. e. the missing statements are chosen uniformly and independently, and it is time to find a solution. Vasya is interested in two values. The first one is the expected number of solutions. The second is the probability that at least one solution exists.

Unfortunately, Vasya could have made a mistake while recording the statements on the meeting. So, the initial data could be contradictory and, as a result, it may happen that there are no solutions for any of the possible protocols. In this case, the two required values are zeroes.

Recall that the expected value of a random variable is a sum over all possible outcomes (in our case, these are protocols) of the probability of an outcome multiplied by the value of this variable (in our case, the number of solutions) for this outcome.

Input

The first line of input contains two integers n and k separated by a space: the number of inhabitants and the number of statements Vasya heard ($1 \le n \le 50$, $0 \le k \le n$). The next k lines describe the statements. Each of these lines has the form $i\ j\ s$ where i is the number of inhabitant which pronounced this statement ($1 \le i \le n$), j is the number of inhabitant this statement refers to ($1 \le j \le n$), and s is either "`knight`" or "`liar`" which is what inhabitant i said about inhabitant j.

It is guaranteed that all the statements are produced by different inhabitants. Remember that the data could be contradictory.

Output

On the first line of output, write two numbers separated by a space: the expected number of solutions and the probability that at least one solution exists. The answer is considered correct if it differs from the exact one by no more than 10^{-9}.

Example

standard input	standard output
3 2 1 2 liar 3 1 knight	1.0000000000 0.5000000000

Example explanation

In this example, inhabitant 1 says that inhabitant 2 is a liar and inhabitant 3 says that inhabitant 1 is a knight. There are six possible protocols. They all differ only by what inhabitant 2 said. Each of them has a probability of $\frac{1}{6}$. The number of possible solutions is shown in brackets. The solutions themselves are coded by strings of three characters describing inhabitants in natural order (`K` for knight, `L` for liar).

1. Inhabitant 2 says that inhabitant 1 is a knight (0 solutions).
2. Inhabitant 2 says that inhabitant 1 is a liar (2 solutions: `KLK` and `LKL`).
3. Inhabitant 2 says that inhabitant 2 is a knight (2 solutions: `KLK` and `LKL`).
4. Inhabitant 2 says that inhabitant 2 is a liar (0 solutions).
5. Inhabitant 2 says that inhabitant 3 is a knight (0 solutions).
6. Inhabitant 2 says that inhabitant 3 is a liar (2 solutions: `KLK` and `LKL`).

The expected number of solutions is $\frac{1}{6} \cdot 0 + \frac{1}{6} \cdot 2 + \frac{1}{6} \cdot 2 + \frac{1}{6} \cdot 0 + \frac{1}{6} \cdot 0 + \frac{1}{6} \cdot 2 = 1$. The probability that at least one solution exists is $0 + \frac{1}{6} + \frac{1}{6} + 0 + 0 + \frac{1}{6} = \frac{1}{2}$.

Problem L. Lazy Managers

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

The boss of the international company "Horns and Hooves, Inc" O'Bender sent two managers, Alex and Mike, to Moscow office.

It is known that Alex and Mike will stay in Moscow for n days. They are lazy, so each of them may use the underground not more than for one trip per day. The manager's schedule is know to Adam, the manager of Moscow office. Each morning Adam gives to each person an underground ticket if that person needed it today (so when both managers are going to the city, each of them receive a ticket). At the end of day Adam takes the tickets back from managers. Due to special relations with the some people in the administration, the boss of the company may order the tickets for A passages and B days very cheap, so managers are using only this type of tickets.

But Adam wants to save O'Bender's money as much as possible and wants to buy minimum possible number of tickets. Note that Adam may determine which tickets to give Alex and Mike in the morning. Your job is to help Adam and calculate minimum number of tickets needed.

Note that the single ticket allows its owner to use underground no more than A times and the number of days between the first and the last passage should be less than B.

Input

In the first line of the input there is integer n ($1 \leq n \leq 200$) and the integers A and B ($1 \leq A, B \leq 20$). In the second and third lines contain numbers $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots b_n$ respectively ($a_i, b_i \in \{0, 1\}$). Alex uses underground on the i^{th} day if and only if $a_i = 1$. Mike uses underground on the i^{th} day if and only if $b_i = 1$.

Output

Output one number — the minimum number of tickets that O'Bender should order for Alex and Mike.

Examples

standard input	standard output

`2 5 5`
`1 1`
`0 0`
	`1`

`2 5 5`
`1 0`
`0 1`
	`1`

`11 20 10`
`1 1 1 1 1 1 1 1 1 1 1`
`1 0 0 0 0 0 0 0 0 0 0`
	`3`

Note

In the last sample Adam give two tickets to the Alex and Mike at the first day, so they stop working at the eleventh day and he should give to Alex one more ticket.

Problem M. Matches Arrangement

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

You are given a string r, consisting of Roman numerals (‘`I`’, ‘`V`’, ‘`X`’, arithmetic operators (‘`+`’, ‘`-`’) abd the equation sign (‘`=`’). Each character is represented by one (‘`I`’, ‘`-`’) or two (all other) matches. Your task is to move the minimum number of matches to obtain correct equality.

Note that:

- Spaces between characters are big enough, so there is no limit on inserting matches or its combinations between characters.
- The ‘`+`’ and ‘`=`’ after removal of one match turns into ‘`-`’ and vice versa, ‘`-`’ can be turned to any of those characters by adding one match; ‘`+`’ can be converted into ‘`=`’ by moving exactly one match
- ‘`V`’ can be transformed to ‘`X`’ by moving one match and vice versa.
- The matches from adjanced characters does not forn new character automatically, i.e. it is impossible to remove 2 matches from “`XX`” and obtain ‘`V`’.
- Slope **does matter**, i.e. it is impossible to remove one match from ‘`X`’ and ‘`V`’ to obtain ‘`I`’
- The unary ‘`+` and ‘`-`’ in the answer are not allowed. The answer shall include exactly one equation sign ‘`=`’.
- The rules for Roman numerals are contemporary; namely, for numbers like 4, 9 etc cannot be used representation with four similar digits in a row (i.e. ‘`IIII`’ or ‘`XXXX`’ is not allowed).
- The number of matches in the input and output expressions must be exactly the same.

Input

The input contains of one non-empty string, containing match-build text. Number of used matches in this text does not exceed 13.

Output

In first line of the output, print -1 if it is impossible to solve given puzzle. Otherwise in the first line print minimal number of matches needed to be moved to solve given puzzle, and in second — resulting equation. If there are more than one solutions, print any of them.

Examples

standard input	standard output

`IV=XI+V`	`1`
	`IV=X-I-V`

`X`	`-1`

`XXIVI`	`3`
	`VI=VI`