Problem A. ASCII Puzzle

Input file: `ascii.in`
Output file: `ascii.out`

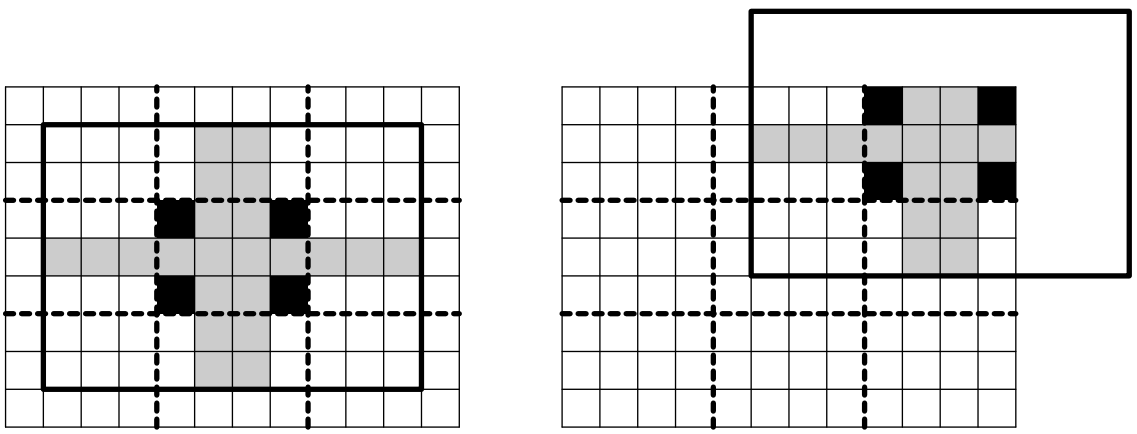
Fili and Floi play a puzzle game. Fili takes a rectangular piece of paper that is lined with a $W \times H$ grid of square cells, cuts it into pieces on its grid lines, and carefully shuffles the pieces so that pieces do not rotate. Floi has to recombine the pieces back into the rectangle without rotating them.

Fili observes a number of constraints while cutting an original paper into pieces to make sure that the resulting puzzle is well-formed. First of all, Fili picks three integer numbers w, h, and n, so that an original rectangular paper has a width of $W = wn$ cells and a height of $H = hn$ cells. Here w and h are known to Floi, but n, W, and H are not. This way, the original rectangular piece of paper can be cut into a trivial puzzle of $k = n^2$ rectangles with a width of w cells and a height of h cells each. However, this trivial puzzle for $k > 1$ is not considered a well-formed puzzle for this game. Instead, the pieces into which the original rectangle is cut are based on these trivial $w \times h$ cell rectangles with the jagged edges between the adjacent pieces. Formally, the pieces into which the original $W \times H$ paper is cut satisfy the following constraints of a *well-formed puzzle*:

- There are $k = n^2$ pieces.
- Each piece is a simple 4-connected region of cells without holes.
- Each cell of the original rectangular $W \times H$ paper is a part of exactly one piece.
- Each piece contains four corners of the corresponding $w \times h$ rectangle in the trivial puzzle for the original paper.
- The cells of each piece can come only from the corresponding $w \times h$ rectangle in the trivial puzzle, from the cells adjacent to this rectangle, and from the interior cells of the adjacent rectangles in the trivial puzzle.
- The cut between two adjacent pieces cannot be straight. Only pieces that lie on the border of the original $W \times H$ paper have straight sides.

The corollary of these constraints is that each piece of a well-formed puzzle fits into a rectangle of $(3w - 2) \times (3h - 2)$ cells. Moreover, the description of each piece will be given as a $(3w - 2) \times (3h - 2)$ grid of cells in such a way, that the corresponding $w \times h$ rectangle of the trivial puzzle is exactly in the center.

The picture below to the left shows a sample rectangular piece of paper that is lined with a $W \times H = 12 \times 9$ square grid of cells and is cut into a trivial puzzle of $k = 9$ rectangles with a width of $w = 4$ cells and a height of $h = 3$ cells each with bold dashed lines. The corners of the central 3×4 piece of this trivial puzzle are shown in black. They have to be a part of the central piece of any well-formed puzzle. The other potential cells of the central piece of a well-formed puzzle are shown in gray. The bold black line shows $(3w - 2) \times (3h - 2) = 10 \times 7$ rectangular region that will be describing this central piece. The picture to the right shows the same for the piece in the upper-right corner of the puzzle.



Your task is to help Floi solve the puzzle.

Input

The first line of the input file contains there integers k, w and h. Here k is the number of pieces in the puzzle, w is a width and h is a height of a trivial puzzle piece ($k = n^2$ for $1 \le n \le 4$, $3 \le w, h \le 5$).

The rest of the input file contains descriptions of k pieces of a well-formed puzzle. Each piece is described by $3h - 2$ lines that contain $3w - 2$ characters each. Pieces are labeled with a consecutive English letters in uppercase (1st piece — '`A`', 2nd piece — '`B`', and etc). Each piece description uses only two characters on its $3h - 2$ lines of $3w - 2$ characters. The English letter corresponding to the piece denotes a cell that is a part of this piece, while '`.`' (dot) character denotes a cell that is not.

Empty lines separate different pieces.

Output

The first line of the output file shall contain W and H — the size of the original piece of paper that was cut into the puzzle pieces. The following H lines shall contain W English letters each, describing the solution of the puzzle. Letters denote the cells that belong to the corresponding puzzle pieces. If there are multiple ways to solve the puzzle, then print any solution.

Sample input and output

ascii.in	ascii.out
4 4 3 ....AAAA... ...AAAAAA. ...A.AA... ....BBBB... .....BB... ...BBBB... ....BB.... .....B.... ....C..C... ..CCC.C... ...CCCC... ....D..... ...DDDD... ...DDD.... ...DDDD...	8 6 AAAABBBB AAAAAABB ADAABBBB DDDDCBBC DDDCCCBC DDDDCCCC

Problem B. Bonus Cards

Input file: `bonus.in`
Output file: `bonus.out`

Dmitry loves programming competitions very much. The Finals of the famed Champions League are taking place in Dmitry's home city, so he wants to visit the competition. The competition is very popular, but most tickets to the competition are reserved for VIPs and for sponsors.

For the general public tickets to the Champions League Finals are distributed in the following way. Spectators, that want to see the competition, submit their request that states the payment method they want to use to pay for their ticket. Suppose there are n seats available. Several draw round are conducted. In each round every request that is not yet fulfilled receives some number of slots depending on the payment method. Then one of those slots is selected uniformly at random. The request to which this slot belongs is considered fulfilled and does not take part in subsequent drawing rounds. Draw ends after n rounds or when there are no more unfulfilled requests, whichever occurs first. An International Card Processing Corporation (ICPC) is a sponsor of the Champions League. Those who chose ICPC card as a payment method receive two slots in each draw round, while users of other payment methods receive only one.

Dmitry has a card from ICPC, but he also has a card from Advanced Credit Merchandise (ACM), which offers him a bonus on all his spendings. His brother Petr works in a company that conducts draw to distribute tickets, so he told Dmitry in advance how many people had already decided to use ICPC card and how many had decided to use other methods. Now Dmitry want to know the probabilities he would get a ticket if he would use his ICPC card or if he would use his ACM card, so that he can make an informed choice. His request is going to be in addition to the number of requests Petr had told him about.

Can you help?

Input

The first and the only line of the input contains 3 integer numbers — the number of seats available for a draw n ($1 \le n \le 3000$), the number of requests with ICPC card as a payment method a, and the number of requests with other payment methods b ($0 \le a, b \le 10^9$).

Output

On the first line output the probability of getting a ticket using ICPC card. On the second line output the probability of getting a ticket using ACM card. Answers should have an absolute error of no more than 10^{-9}.

Sample input and output

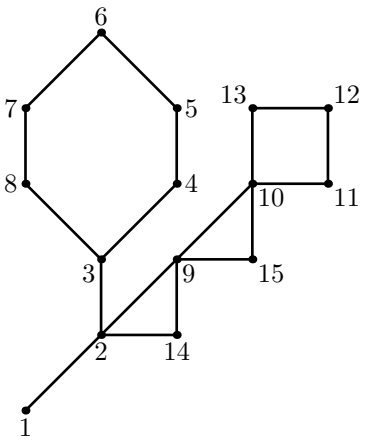
bonus.in	bonus.out
1 1 2	0.3333333333333333 0.2
10 10 10	0.5870875690480144 0.3640355515319861

Problem C. Cactus Automorphisms

Input file: `cactus.in`
Output file: `cactus.out`

NEERC had featured a number of problems in previous years about *cactuses* — connected undirected graphs in which every edge belongs to at most one simple cycle. Intuitively, cactus is a generalization of a tree where some cycles are allowed.

In 2005, the first year where problems about cactuses had appeared, the problem was called simply "Cactus". In 2007 it was "Cactus Reloaded" and in 2010 it was called "Cactus Revolution". An example of cactus from NEERC 2007 problem is given on the picture below.



The challenge that judges face when preparing test cases for those problems is that some wrong solutions may depend on the numbering of vertices in the input file. So, for the most interesting test cases judges typically include several inputs with the same graph, but having a different numbering of vertices. However, some graphs are so regular that the graph remains the same even if you renumber its vertices. Judges need some metric about the graph that tells how regular the given graph is in order to make an objective decision about the number of test cases that need to be created for this graph.

The metric you have to compute is the number of graph *automorphisms*. Given an undirected graph (V, E), where V is a set of vertices and E is a set of edges, where each edge is a set of two distinct vertices $\{v_1, v_2\}$ $(v_1, v_2 \in V)$, graph automorphism is a bijection m from V onto V, such that for each pair of vertices v_1 and v_2 that are connected by an edge (so $\{v_1, v_2\} \in E$) the following condition holds: $\{m(v_1), m(v_2)\} \in E$.

Each graph has at least one automorphism (one where m is an identity function) and may have up to $n!$ automorphisms for a graph with n vertices. Because the number of automorphisms may be a very big number, the answer must be presented as a prime factorization $\prod_{i=1}^{k} p_i^{q_i}$, where p_i are prime numbers in ascending order ($p_i \geq 2$, $p_i < p_{i+1}$) and q_i are their corresponding powers ($q_i > 0$).

Input

The first line of the input file contains two integer numbers n and m ($1 \leq n \leq 50\,000$, $0 \leq m \leq 50\,000$). Here n is the number of vertices in the graph. Vertices are numbered from 1 to n. Edges of the graph are represented by a set of edge-distinct paths, where m is the number of such paths.

Each of the following m lines contains a path in the graph. A path starts with an integer number k_i ($2 \leq k_i \leq 1000$) followed by k_i integers from 1 to n. These k_i integers represent vertices of a path. Adjacent vertices in a path are distinct. Path can go to the same vertex multiple times, but every edge is traversed exactly once in the whole input file. There are no multiedges in the graph (there is at most one edge between any two vertices).

The graph in the input file is a cactus.

Output

On the first line of the output file write number k — the number of prime factors in the factorization of the number of graph automorphisms. Write `0` if the number of graph automorphisms is 1. On the following k lines write prime numbers p_i and their powers q_i separated by a space. Prime numbers must be given in ascending order.

Sample input and output

cactus.in	cactus.out
15 3 9 1 2 3 4 5 6 7 8 3 7 2 9 10 11 12 13 10 5 2 14 9 15 10	1 2 2
2 1 2 1 2	1 2 1
15 7 3 1 2 3 3 4 2 5 3 6 2 7 3 8 2 9 3 10 2 11 3 12 2 13 3 14 2 15	6 2 11 3 5 5 2 7 2 11 1 13 1

The first sample input corresponds to the picture from the problem statement. This graphs has $4 = 2^2$ automorphisms.

The second sample input is a simple graph with two vertices and one edge between them that has $2 = 2^1$ automorphisms.

The third sample input is a "star" graph with a center vertex and 14 rays that has $14! = 87\,178\,291\,200 = 2^{11} \times 3^5 \times 5^2 \times 7^2 \times 11^1 \times 13^1$ automorphisms.

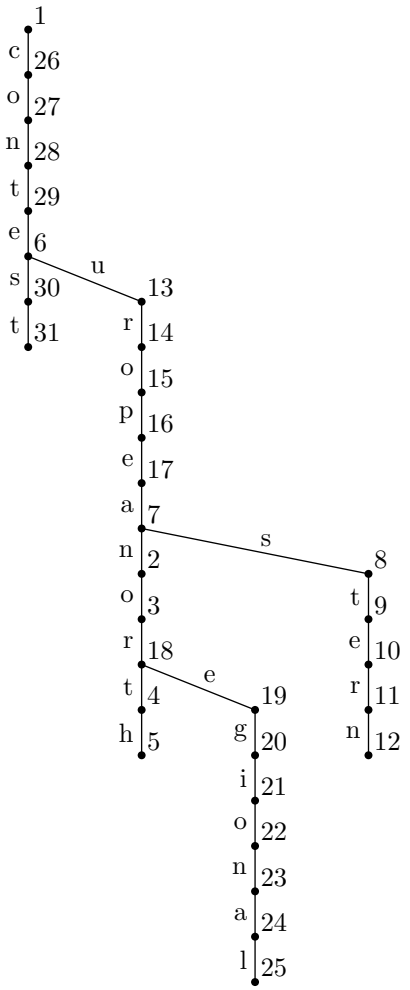
Problem D. Dictionary

Input file: `dictionary.in`
Output file: `dictionary.out`

Petr and Dmitry are working on a novel data compression scheme. Their task is to compress a given set of words. To compress a given set of words they have to build a rooted tree. Each edge of the tree is marked with exactly one letter.

Let us define a *dictionary* that is produced by this kind of tree as a set of words that can be constructed by concatenating letters on edges on any path from any vertex in the tree (not necessarily root) and going away from root down to the leaves (but not necessarily finishing on a leaf).

Boys have to construct such a tree with a dictionary that is a superset of the set of words that they are given to compress. This tree should have the smallest number of vertices between trees that satisfy the above condition. Any tree with the same number of vertices will do. Your task is to help them.



For example, in a tree on the picture above with the root marked as 1, a path from 7 to 5 reads "north", a path from 16 to 12 reads "eastern", a path from 29 to 2 reads "european", a path from 3 to 25 reads "regional", and a path from 1 to 31 reads "contest".

Input

The first line of the input file contains the number of words in a given set n ($1 \le n \le 50$). The following n lines contain different non-empty words, one word per line, consisting of lowercase English letters. The length of each word is at most 10 characters.

Output

On the first line output the number of vertices in the tree m. The following m lines shall contain descriptions of tree vertices, one description per line. Vertices are indexed from 1 to n in the order of their corresponding description lines. If the corresponding vertex is a tree root, then its description line shall contain a single integer number 0, otherwise its description line shall contain an index of its parent node and a letter on the edge to its parent node, separated by a space.

Sample input and output

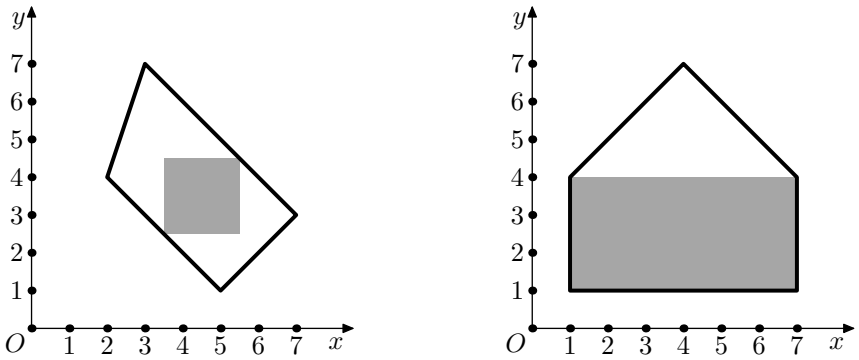
dictionary.in	dictionary.out
5 north eastern european regional contest	31 0 7 n 2 o 18 t 4 h 29 e 17 a 7 s 8 t 9 e 10 r 11 n 6 u 13 r 14 o 15 p 16 e 3 r 18 e 19 g 20 i 21 o 22 n 23 a 24 l 1 c 26 o 27 n 28 t 6 s 30 t

This sample output corresponds to the picture from the problem statement.

Problem E. Easy Geometry

Input file: `easy.in`
Output file: `easy.out`

Eva studies geometry. The current topic is about convex polygons, but Eva prefers rectangles. Eva's workbook contains drawings of several convex polygons and she is curious what is the area of the maximum rectangle that fits inside each of them.



Help Eva! Given the convex polygon, find the rectangle of the maximum possible area that fits inside this polygon. Sides of the rectangle must be parallel to the coordinate axes.

Input

The first line contains a single integer n — the number of sides of the polygon ($3 \le n \le 100\,000$).

The following n lines contain Cartesian coordinates of the polygon's vertices — two integers x_i and y_i ($-10^9 \le x_i, y_i \le 10^9$) per line. Vertices are given in the clockwise order.

The polygon is convex.

Output

Output four real numbers x_{min}, y_{min}, x_{max} and y_{max} — the coordinates of two rectangle's corners ($x_{min} < x_{max}$, $y_{min} < y_{max}$). The rectangle must fit into the polygon and have the maximum possible area.

The absolute precision of the coordinates should be at least 10^{-5}.

The absolute or relative precision of the rectangle area should be at least 10^{-5}. That is, if A' is the actual maximum possible area, the following must hold: $\min(|A - A'|, \frac{|A-A'|}{A'}) \le 10^{-5}$.

Sample input and output

easy.in	easy.out
4 5 1 2 4 3 7 7 3	3.5 2.5 5.5 4.5
5 1 1 1 4 4 7 7 4 7 1	1 1 7 4

Problem F. Fraud Busters

Input file: `fraud.in`
Output file: `fraud.out`

The number of cars in Default City that travel to the city center daily vastly exceeds the number of available parking spots. The City Council had decided to introduce parking fees to combat the problem of overspill parking on the city streets. Parking fees are enforced using an automated vehicle registration plate scanners that take a picture of the vehicle registration plate, recognize the sequence of digits and letters in the code on the plate, and check the code against a vehicle registration database to ensure that parking fees are dutifully paid or to automatically issue a fine to the vehicle owner otherwise.

As soon as parking fees were introduced, a parking fee fraud had appeared. Some vehicle owners had started to close one or several digits or letters on their vehicle registration plate with pieces of paper while they park, thus making it impossible for the current version of the automated scanner to recognize their vehicle's registration code and to issue them a fine.

The Default City Council had instituted the Fraud Busters Initiative (FBI) to design a solution to prevent this kind of fraud. The overall approach that FBI had selected is to expand the number of vehicle features that scanners recognize (including features like vehicle type and color), as well as excluding from the list any vehicles that are detected to be elsewhere at this time. This information should help to identify the correct vehicle by narrowing down the search in the vehicle registration database.

You are working for FBI. Your colleagues had already written all the complex pieces of the recognition software that analyses various vehicle features and provides you with a list of registration codes that might potentially belong to a scanned car. Your task it to take this list and a recognized code from the license plate (which may be partially unrecognized) and find all the registration codes that match.

Input

The first line of the input file contains 9 characters of the code as recognized by the scanner. Code that was recognized by the the scanner is represented as a sequence of 9 digits, uppercase English letters, and characters "`*`" (star). Star represents a digit or a letter that scanner could not recognize.

The second line of the input file contains a single integer number n ($1 \le n \le 1000$) — the number of vehicle registration codes from the vehicle registration database.

The following n lines contain the corresponding registration codes, one code per line. Vehicle registration codes are represented as a sequence of 9 digits and uppercase English letters. All codes on these n lines of the input file are different.

Output

On the first line of the output file write a single integer k ($0 \le k \le n$) — the number of codes from the input file that match the code that was recognized by the scanner. The code from the scanner matches the code from the database if the characters on all the corresponding positions in the codes are equal or the character from the scanner code is "`*`".

On the following k lines write the matching codes, one code per line, in the same order as they are given in the input file.

Sample input and output

fraud.in	fraud.out
A**1MP19* 4 A001MP199 E885EE098 A111MP199 KT7351TTB	2 A001MP199 A111MP199

Problem G. Green Energy

Input file: `green.in`
Output file: `green.out`

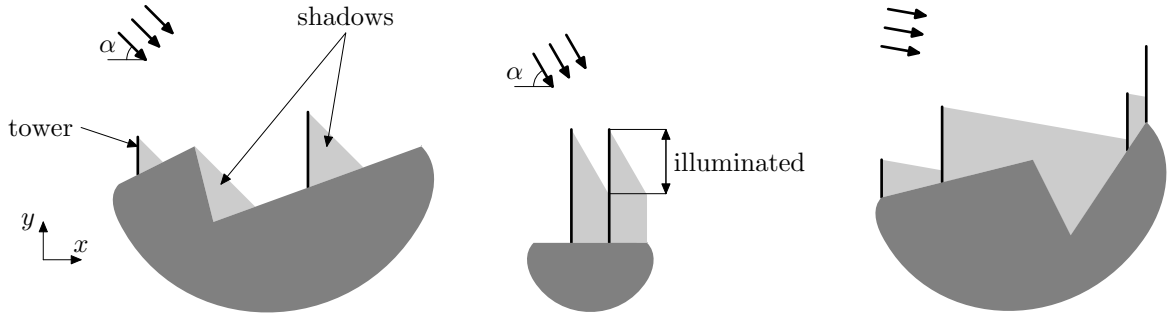
The technological progress in Flatland is impressive. This year, for example, the solar power stations of a new type will be build. In these stations solar panels are mounted not on the ground, but on high towers, along their heights.

There are n towers to be mounted. The towers are already bought. The height of i-th tower is h_i. Now engineers want to choose the points where they should be mounted to get the maximal total power.

The landscape of a territory of the power plant is described by a polyline with m vertices. Vertices of the landscape polyline have coordinates (x_i, y_i), such that $x_i < x_{i+1}$.

The sun angle is always α degrees in Flatland. The sun is shining from the top-left to the bottom-right. The power that is produced by a tower depends on the length of its surface illuminated by the sun.

When two towers are mounted close to each other, the shadow of the left tower may fall onto the right tower, so the power, produced by the right tower, decreases. Also, the landscape itself may contain high points that drop shadows on some towers.



Your task is to find the points on the territory of the plant to mount the given towers to maximize the total length of towers surface that is illuminated by the sun.

Input

The first line of the input file contains three integers n, m and α ($1 \le n \le 10^4$, $2 \le m \le 10^4$, $1 \le \alpha < 90$). The second line contains n integers h_i — the heights of the towers ($1 \le h_i \le 10^3$). The following m lines contain pairs x_i, y_i — the coordinates of the vertices of the landscape ($|x_i| \le 10^5$, $x_i < x_{i+1}$, $|y_i| \le 10^3$).

Output

On the first line output the maximal possible summary length of towers that can be illuminated by the sun with an absolute precision of at least 10^{-6}. On the next n lines output the x-coordinates of the points where the towers should be mounted to achieve this maximum with an absolute precision of at least 10^{-9}. Towers should be listed in the same order they are given in the input file.

Sample input and output

green.in	green.out
5 4 10 20 10 20 15 10 0 10 40 20 50 0 70 30	52.342888649592545 16.0 0.0 70.0 65.3 65.3

In this example two towers are mounted at the same point. This is allowed, but only one, the longest, of the towers mounted at the same point is considered to be illuminated by the sun.

Problem H. Hack Protection

Input file: `hack.in`
Output file: `hack.out`

Pavel is sending to his friend Egor some array of non-negative integers. He wants to be sure, that nobody hacks the array before his friend gets it. To solve this problem Pavel need to compute some kind of a checksum or a digest for his array. Pavel has an innovative mind, so he invents the following algorithm to compute the digest for his array: count the number of subarrays in which the bitwise *xor* of the numbers in the subarray is equal to the bitwise *and* of the same numbers.

For example, consider an array of four binary numbers "01", "10", "11", and "11". The table below to the left lists the results of the bitwise *xor* of numbers for each subarray of this array, and the table below to the right list the results of the bitwise *and* of numbers for each subarray of this array. The rows of the table correspond to the starting elements of the subarray from the 1st element of the array to the 4th one, while columns correspond to the ending elements of the subarray. Matching values are highlighted with gray background.

xor

01	11	00	11
	10	01	10
		11	00
			11

and

01	00	00	00
	10	10	10
		11	11
			11

Your task is to help Pavel compute this kind of a digest of the given array.

Input

The first line contains one integer n ($1 \le n \le 100\,000$). The second line contains n non-negative integers a_i ($0 \le a_i \le 2^{31} - 1$) that are written in decimal notation.

Output

On the first line of the output print Pavel's digest of the given array.

Sample input and output

hack.in	hack.out
4 1 2 3 3	6

The above sample input corresponds to the example from the problem statement.

Problem I. Interactive Interception

Input file: `standard input`
Output file: `standard output`

This is an interactive problem.

North Eastern Emergency Rocket Control agency (NEERC) has developed a new radar control system that is designed to better control ballistic rocket interception. To test the new system NEERC agency had developed a mathematical model that is intended to show this system's abilities.

Let us represent a rocket as a point on a line. Initially the point is at some unknown integer location between 0 and p, inclusive. It has some unknown speed of q which is an integer between 0 and v, inclusive.

Each second the following happens. First, the control system makes a query to the radar of a form "`check` L R" and gets an answer whether the point is currently between L and R, inclusive, or not. After that, the point's coordinate increases by q.

The goal of the radar control system is to learn the exact location of the point at the beginning of some second. When it does learn the point's location, then instead of making a query to the radar, it gives a command to intercept the point at that location.

You have to implement the control system that locates and intercepts the point while making at most 100 queries to the radar.

Interaction protocol

Interaction starts with your program reading two integers — the values of p and v from the standard input ($1 \le p \le 10^5$, $1 \le v \le 10^5$).

After that your program must print commands to the standard output. Each command must be one of the following two.

- "`check` L R" — make a query to the radar to get an answer whether the point is currently between L and R, inclusive, or not. The answer must be read from the standard input and is either "`Yes`" or "`No`". After that the point's coordinate is increased by q. L and R must be integers ($0 \le L \le R \le 10^9$).

  There must be at most 100 "`check`" commands.

- "`answer` x" — the exact coordinate x of the point is known, and you order to intercept the point. After printing this command your program must exit.

Your program must write end-of-line sequence and flush the standard output after each command, including the last command "`answer` x" (end-of-line must be written and flushed before exiting).

Sample input and output

standard input	standard output
2 2 Yes No Yes Yes	check 1 3 check 3 5 check 2 4 check 4 5 answer 5

In the given example the point was initially at location 1 and is moving at a speed $q = 1$.

Problem J. Join the Conversation

Input file: `join.in`
Output file: `join.out`

Abstract Communication Mastership (ACM) is a software company that develops a unique social network called tWinter.

Each tWinter user has a handle that starts with a commercial at (‘`@`’) character. Users of tWinter social network publish short messages to the network.

If a user's message contains another user's handle (preceded by a space or at the beginning of the message, and followed by a space or at the end of the message) then it is called a *mention*.

A sequence of messages is called a *conversation* if each message in the sequence (except the first one) contains a mention of the author of the previous message in the sequence.

You are hired to find the longest conversation in the given chronological log of messages.

Input

The first line of the input file contains an integer n ($1 \le n \le 50\,000$) — the number of messages in the chronological log.

Each of the next n lines contains a message preceded by its author's handle, a colon (‘`:`’) character, and a space.

Each message is at most 139 characters long. Each handle is at most 20 characters long and does not contain colons or spaces.

The input file contains only characters with ASCII codes between 32 and 126, inclusive, and line breaks.

Output

On the first line of the output file write the length of the longest conversation in the given log. On the second line write 1-based indices of the messages in that conversation in ascending order.

If there are multiple longest conversations, write any one of them.

Sample input and output

join.in	join.out
6 @Petr: Leaving for #NEERC tomorrow! @Roman: This #NEERC is going to be awesome! @Stone_in_forest: Nothing happened today. @NEERCNews: @Petr Don't forget an umbrella :) @Lydia: @NEERCNews cares about @Petr - so cute ^_^ @Lydia: @Lydia @NEERCNews @Petr it won't be raining though!	3 1 4 5

Problem K. Kabaleo Lite

Input file: kabaleo.in
Output file: kabaleo.out

Kabaleo Lite is a board game. The board consists of several stacks of conical chips of various colors. Only the color of the top chip of the stack is visible.

Each player has a unique *target color* and a set of colored chips. The target color is hidden from other players, while the set of chips is visible to them. On his turn, player selects one of his chips and puts it on one of the board stacks, thus recoloring it to the color of the chosen chip.

After the last turn, the number of visible board chips of each color is calculated. The winning color is the color that occurs the maximum times. The player (if any) that has this color as his target color, wins the game. If there is no such player or if there are two or more colors that occur the maximum times, the game ends in a draw.

You are playing your last chip in the Kabaleo Lite game. Other players also have one chip left. You want to determine all possible moves that lead you to winning the game. You do not know the target colors of other players and you cannot predict their moves, so your move must guarantee your victory regardless of moves of your opponents.

Input

The first line contains four integers n, p, c and h — the number of stacks on the board ($1 \le n \le 10^6$), the number of players ($1 \le p \le 10^6$), the number of chips' colors ($p \le c \le 10^6$), and your hidden color ($1 \le h \le c$).

The second line contains n integers b_i — the color of the visible board chip for each stack on the board ($1 \le b_i \le c$).

The third line contains p integers l_i — the color of the last chip for each player ($1 \le l_i \le c$). The players are numbered from one (you) to n in the order of their turns.

Output

The first line must contain w — the number of winning moves.

The second line must contain w distinct numbers m_i — the numbers of the stacks on which your chip should be put to win. Stacks are numbered starting from 1 in the order that their visible colors are given in the input file. You can output their numbers in any order on this line.

Remember, that your move should be winning regardless of the moves of all other players.

Sample input and output

kabaleo.in	kabaleo.out
6 3 4 2 2 1 2 3 2 2 2 1 1	1 2

Note, that if you put the chip on the 4th stack, other players may place their chips on the 1st and the 3rd stack, which leads to a draw, because the number of visible chips of the first and the second colors is the same (three chips of each color).