Problem A. Ministry

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

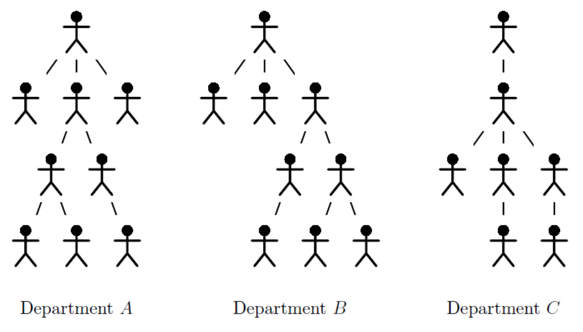
Once upon a time in a country far, far away, the government established the Ministry of paperwork reduction. As you might have guessed, it was the largest ministry ever. The number of officials working there was truly enormous. However, the overall structure was very simple: The minister had at most three subordinate officials, each of them again had at most three subordinate officials and so on. The recent elections have brought a new minister into this office. He is young, smart and full of unspoilt ideals. He decided to fulfill the name of his institution and start at his doorstep. He observed that many parts of the hierarchical structure of the ministry are the same and therefore they must be doing the same job. And whenever two such units do the same, one of them is superfluous and it can be dissolved and all its officials laid off. Your job is to find how many non-equivalent departments there are and to fill in the necessary paperwork (i.e., write the result to the output in the correct format).

You are given the organizational structure of the ministry. Each official has exactly one superior official and at most three subordinate officials (including, possibly, zero). The only exception is that the minister doesn't have any superior (but he is still limited to three subordinates). There is no specific order of subordinates of any individual official. A department consists of an official, all his subordinates, all their subordinates and so on. There are two special cases of departments: the full ministry (starting with the minister) and one-man departments consisting of a single official who has no subordinates.

The depth of a department is the length of the longest chain $x_1, \ldots, x_d$ of the officials in the department such that x_i is the superior of x_{i+1} for every $1 \le i < d$. Observe that the depth of a one-man department is 1.

Two departments A and B have the same structure if each official x from department A corresponds to a unique official x' from department B and vice versa each official x' of B corresponds to an official x of A. In particular, the following must hold for all officials x and y: x is the superior of y if and only if x' (the official corresponding to x) is the superior of y' (the official corresponding to y). Observe that if the departments A and B have the same structure, then the head of the department A corresponds to the head of B and both the departments have the same depth and the same number of officials.

In the following picture, the departments A and B have the same structure, while the structure of the department C is different from both A and B:



Your task is to determine the number of departments with different structure for all depths. In other words, you have to produce a sequence $n_1, \ldots, n_d$ such that d is the depth of the whole ministry and for each i the ministry contains exactly n_i departments of depth i with pairwise different structures.

Input

The input consists of a single line which describes the organizational structure of the ministry using the following notation. Each department is encoded as $(x_1, \ldots x_k)$ where $0 \le k \le 3$ is the number of the head's subordinates and the x_i's are the codes of their departments. A one-man department is therefore encoded as (). The whole structure of the ministry is described as the code of the full ministry.

The ministry contains at most 1 000 000 officials (including the minister).

Output

The output should consist of d lines, where d is the depth of the ministry, i.e., the depth of the department led by the minister. The i-th line contains the number of departments of depth i with different structure.

Example

ministry.in
`((())())((()())(()()()))(()(()))`
ministry.out
1 3 2 1

Problem B. Nasty Calculations

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

Your friend Joe forgot to write his math homework, and his teacher got angry (guess why). He ordered Joe to stay at school after the classes and evaluate a long list of expressions. The teacher is, however, quite lazy to think up so many different expressions, therefore he gave Joe a single formula f of a single variable x and a large number of values for x (as you understand, those are much easier to generate). Joe's task is to compute the value of $f(x)$ for each of the given values of x.

In fact, Joe is not required to compute the exact value of $f(x)$, since it might be quite large. Instead, he should write only the last digit of the value of $f(x)$ (Joe thinks that the reason is that the teacher can count only up to 100, but who knows...). Since the topic of the last class were numeral systems with different bases, all calculations and expressions (in particular, the expression describing f, the values for x, and the corresponding last digits of $f(x)$) are written in the B-base numeral system, where B is a given integer.

As the number of values for x is really large and the formula is by no means short, Joe asked you to help. He knows that you are an excellent programmer, thus he thinks you could use the computer to solve the problem. In order to make your task simpler, he has rewritten the formula f to the postfix notation (explained below), since he is aware of the fact that it is easier for computers to evaluate such formulas. Your task is to determine the last digit of the value of $f(x)$ for the given formula f and for given values of x in the B-base numeral system.

The postfix notation (also known as the Reverse Polish notation, RPN) is an alternative way of writing mathematical expressions.

Using the most common infix notation, an operator is placed between its operands, such as $1+2$, $1+2*3$, or $(1+2)*3$. Unlike that, using the postfix notation, the operator is placed directly after its operands: the above expressions are encoded as $12+$, $123**+$, and $12+3*$.

Input

The first line of the input contains exactly two integers — B and N separated by a single space. B $(2 \le B \le 36)$ is the base of the numeral system and N $(1 \le N \le 100\,000)$ is the number of values for x given in the input.

The second line contains the description of the formula f in the postfix notation — it consists of several elements separated by single spaces. Those elements can be:

- A sequence of digits and upper case letters. Such a sequence describes a number written in a base-B numeral system as described earlier. You may assume that the represented number will not exceed $2\,000\,000\,000$.
- The lower case letter x. This symbol should be replaced with the appropriate value of x in every calculation.
- The addition operator $+$.
- The subtraction operator $-$.
- The multiplication operator $*$.

Each of the next N lines contains a single sequence of digits and upper case letters which encodes the corresponding value for x written in the base-B system in the same fashion as above. Again, you may assume that none of the values exceeds $2\,000\,000\,000$.

You may also expect that the input data is correct, i.e., the content of the second line is a valid expression written in the postfix notation and every sequence of digits and upper case letters is a valid integer in the base-B notation. Further, you may assume that the length of the second line will not exceed $100\,000$ characters.

Output

The output should contain exactly N lines. The i-th line should then contain exactly one character, either a digit or an upper case letter that is the last digit of the value of $f(x)$ (in the base-B numeral system) for x given by the $(i+2)$-th line of the input. You may assume that the value of $f(x)$ will be non-negative for all values of x given in the input.

Example

nasty.in	nasty.out
15 4 2 x * 123A + 1 2 3 4	C E 1 3

Problem C. Airport Show

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

The management of the airline company Flying Bugs decided to prepare an aerial show as a part of their advertising campaign. The show will take a place at the company airport which has N runways (numbered $1, \ldots, N$) and will consist of performances of two airplane acrobatic groups held in parallel. Each performance has a predefined program and to keep its visual effect the planes have to take off and land on specific runways in a prescribed order.

The security system of the airport tracks which runways are in use and it will not let any other plane enter the runway during that time. Each performance is delivered to the security team as a sequence of reservations and releases of specific runways. Unfortunately, it cannot be estimated in advance when the requests for reserving and releasing runways will actually appear. The airport authorities would now like to know whether there is a danger that the performances could be delivered in a such a way that at some point neither of the two performances can continue without violating the order of the uses of the runways.

Your task is to write a program that checks whether the requests of the two performances can appear in such an order that neither of the performances can continue. For each of the two performances, you are given a list of requests on reservations and releases of runways. The order of the reservations and releases for each of the two performances satisfies the following conditions:

- a reserved runway is not reserved until it has been released before,
- only a runway previously reserved and not yet released is released, and
- all reserved runways have been released before the end of the performance.

On the other hand, if a certain runway is needed for one of the performances and it is currently reserved for the other one, the former performance can be suspended until the latter one releases the required runway, i.e., the latter performance can reserve and release several runways during its course and eventually release the requested runway by the former performance.

Still, the performances can "block" the airport, if one of them is requiring reservation of runway A while keeping the runway B reserved and the other one is requiring reservation of runway B while keeping runway A reserved. If this happens, neither of the two performances can continue without violating its order of the take-offs and landings. Note that your task is not to check whether the performances can be scheduled in such a way that the airport does not get blocked.

Input

The first line of the input contains a single integer N, $1 \le N \le 1000$, that determines the number of runways at the airport. The rest of the input is comprised of two blocks, each describing one of the performances. The first line of the block contains a single even integer L, $2 \le L \le 5000$, that determines the number of reservations and releases of runways done during the course of the performance. Each of the following L lines contains a 3-character string RES or REL and an integer A, $1 \le A \le N$, which are separated by a single space. The lines starting with the string RES represent demands on reservations of the runways and those starting with REL on releases of the runways. The integer A then determines the number of the runway to be reserved or released.

Output

If the performances always finish without blocking the airport (whatever the time differences between reserving and releasing runways are), the output should contain a single line with the sentence `"The performances will always finish."`.

If there exists a sequence of reservations and releases leading to blocking the airport, the output should contain a description of a possible course of the show leading to such a situation (note that there could be more such courses and you should output an arbitrary one of them). The description of the course of the show should be a sequence $a_1, \dots, a_k$ of the numbers 1 and 2 (with no separating spaces between the numbers). These numbers represent the order in which the demands of the two performances happen (the numbers 1 and 2 represent the first and the second performance): at the beginning, the first of the demands of the a_1-th performance is done, then the first demand of the a_2-th performance if $a_2 \neq a_1$ or the second demand of the a_2-th performance if $a_1 = a_2$, etc. The sequence must be a valid sequence of operations, i.e., at each time, every runway can be reserved by at most one of the two performances. Since the sequence describes the course of the show leading to blocking the airport, the next demand of the first performance which is not covered by the sequence should request reserving a runway reserved for the second performance and the next demand of the second performance should request reserving a runway reserved by the first one.

Example

airport.in	airport.out
2 4 RES 1 RES 2 REL 1 REL 2 4 RES 2 RES 1 REL 2 REL 1	12
4 8 RES 1 RES 2 RES 3 REL 3 REL 2 RES 2 REL 1 REL 2 4 RES 3 REL 3 RES 4 REL 4	The performances will always finish.

Problem D. Royal treasury

Input file: treasury.in
Output file: treasury.out

Once upon a time in a kingdom far far away, the royal treasury started getting emptier and emptier. The king decided to change the situation and he invented a new system of cooperation within the office of the royal treasurer. The clerks of the office are supposed to form pairs (in order to avoid being bribed) in such a way that each pair is formed by a clerk and his/her direct subordinate. Your task is to compute, given the structure of the office of the treasurer, the maximum number of pairs that can be formed this way and in how many different ways this is possible.

The office of the treasurer is led by George Skinflint. Each clerk has zero, one or more subordinates and is a subordinate of a single clerk (except for George Skinflint who is responsible only to the king himself). The number of clerks does not exceed 1000. Your task is to compute the maximum number of pairs that can be formed by clerks in such a way that every pair is formed by a clerk and his/her direct subordinate. In addition, you should also compute the number of ways such pairs can be formed. Note that some clerks need not be contained in a pair.

Input

The first line of the input contains a single number N that represents the number of clerks, $1 \leq N \leq 1000$. The clerks are assigned unique ID numbers from the range between 1 and N. The ID number of the treasurer (Skinflint) is 1. Each of the following N lines corresponds to one of the clerks: it contains his/her ID number, the number K of his/her subordinates, $0 \leq K \leq 999$, and the ID numbers of his/her K subordinates separated by single spaces. You can assume that the line corresponding to a clerk never appears before the line corresponding to his/her supervisor.

Output

The output should consist of two lines. The first line of the output should contain a single number that represents the maximum number M of pairs that the clerks can form. The second line should contain the number of different ways in which the clerks can form M pairs obeying the rules given by the king.

Example

treasury.in	treasury.out

7
1 3 2 4 7
2 1 3
4 1 6
3 0
7 1 5
5 0
6 0
	3
	4

Problem E. Cipher

Input file: cipher.in
Output file: cipher.out

While doing a routine exploration of the Universe, the Balkan Space Agency (BSA) guys have found traces of extraterrestrial (ET) intelligence.

More exactly, they have found a collection of rectangular metal plates containing messages written in the ET's language. Each plate contains a 2D array with n rows and m columns, each element of the array being a printable ASCII character with the corresponding numeric code in the range $32 \dots 127$. Also, each plate contains two more integers, which we will name a and b.

After studying the plates, the specialists of BSA have concluded that the ET's messages can be decrypted by using a cipher, e.g. a key which unveils the way to understand the message, which is hidden right into the rectangular array of that same plate.

The researchers have found out that the cipher is the rectangular subarea of the message, having the size of a rows by b columns that appears exactly k times, $k \geq 3$. The subareas of the message which contain the cipher might overlap. It is also known that no other subarea of a rows by b columns in the metal plate can appear more than $k - 2$ times.

For example, if the rectangular array in the plate has a size of 8×10, e.g. $n = 8$ and $m = 10$, and the cipher appears 5 times on the plate ($k = 5$), and its size is 3×3 ($a = 3$, $b = 3$), no other 3×3 subarray of this array will appear more than 3 times.

You are asked to write a program that, given the rectangular array which represents the ET's message and the integers a and b which are written on the plate, finds the cipher and the positions on the plate where it can be found.

Input

The first line of the input file contains the integers n and m, separated by a blank character. Then n lines follow, each containing one m characters long string. The i-th line represents the i-th line of the rectangular array written on the plate. The last line of the file contains the integers a and b, separated by a blank.

Output

The first line of the output file must contain the integers a and b, separated by a blank. These numbers should be exactly equal to the ones found in the input file. Each of the next a lines of the text should contain b characters long string, representing the cipher you have found. The next line of the output file must contain the integer k, the number of instances of the cipher you have found in the array. Then k lines of text should follow, each containing two integers delimited by a blank. These two numbers must represent the (row, column) position of the upper-left corner of the instances of the cipher you have found in the array. Output pairs in lexicographical order.

Example

cipher.in	cipher.out
8 10 qw.aba..f. wq.bab.ff. zx.cdc.K.R c.ababa.es x.babab.Ed j.cdcdcaba yo.k.k.bab opu..l.cdc 3 3	3 3 aba bab cdc 4 1 4 4 3 4 5 6 8

Problem F. Strange Dream

Input file: dream.in
Output file: dream.out

Dumitru had a very strange dream: he dreamed that he was in a room having its door locked. There he found n boxes each containing exactly m small plates, on each plate being written an integer number greater or equal to 1. He also found a small note containing 2 integer numbers k and l, and specifying the following task:

Step 1: Pick one plate from the first box, write its number in your notebook, change the number on the plate to 1, and put the plate back into the box. Subsequently, in the same way, pick a single plate from the second box, then one from the third box, and so on till the n-th (last) box, inclusive, each time picking a plate from the respective box, writing its number in your notebook, changing the number on that plate to 1 and then placing the plate back into that box.

Step 2: After this, in the same manner, pick a single plate from the box $n - 1$, then one from box $n - 2$, then one from box $n - 3$, and so on till the second box, inclusive, each time picking a plate from the respective box.

To unlock the door of the room and to get out, Dumitru needed to find the number T of different ways of picking plates according to the above scenario, such that the product of the numbers that were written in your notebook is divisible by k. Because T could be extremely large, he needed to return the remainder of T when divided by l.

Dumitru is very confused about this dream and so he tries to find the answer to the above task. Write a program which will help Dumitru to solve the task.

Input

The input file contains 2 integer numbers on the first line: n and m, separated by a single space. Second line contains 2 integer numbers k and l separated by a single space. Then n lines follow, each containing m integer numbers separated by single spaces. The first of these lines contains the m numbers written on the plates found in the first box, the second of these lines contains the numbers written on the plates found in the second box, and so on.

$3 \le n \le 200$, $3 \le m \le 10\,000$, $2 \le k \le 200\,000$, $2 \le l \le 30\,000$. Numbers on the plates in boxes have integer values between 1 and 10^6 inclusive.

Output

The output file must contain one integer number that is the remainder of T when divided by l.

Example

dream.in	dream.out

3 3
12 100
5 2 1
2 1 2
3 7 4
	12

Problem G. Points

Input file: points.in
Output file: points.out

You and your friends have invented an interesting game called "Trivial Points". It's played between 2 players and involves placing points in a 3D space in a special way respecting a certain set of rules. Being very excited about this game, you, together with your friends, have decided to create a computer version in which one of the players is a human and the other is computer.

For the purposes of the development of this game you are responsible to write a program that receives a set of points in a 3D space and computes how many different lines are there so that each contains at least 3 of the given points.

Input

The input file contains on the first line a single integer number n, which is the number of given points. Then n lines follow, each containing the 3 integer coordinates of a point, more specifically the i-th of these n lines contains the integer coordinates x_i, y_i, z_i, separated by single spaces, of the i-th point.

$4 \le n \le 1000$. Coordinates x_i, y_i, z_i are integers not exceeding 10^4 by their absolute values.

Output

The output file must contain on the first line a single integer number that is the number of lines asked in problem's statement.

Example

points.in	points.out

7
1 0 -1
3 4 5
2 2 2
3 3 3
-5 -5 -5
1 1 1
-3 4 0
	2

Problem H. Mokia

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

The Moldovan Mobile Phone Company Mokia has designed a new customer location system. Like any location system, it can answer easily, with millimeter precision, any query of the type "What is the position of the customer C?", but the real advance in technology is the ability to answer to queries of the type "How many customers are there in a given rectangular area?".

In the location system the world is viewed as a square area with a certain size $W \times W$, composed of elementary squares of size 1×1. An elementary square is defined by its (x, y) indices, $1 \leq x, y \leq W$. The indexing starts at 1.

Please help Mokia and write a program that computes how many customers are there in a given rectangular area.

Input

The input is encoded as follows. Each input comes on a separate line, and consists of one instruction integer and a number of parameter integers according to the following table.

Instr	Param	Meaning
0	W	Initialize the matrix size to $W \times W$ containing all zeros. This instruction is given only once and it will be the first instruction. $1 \leq W \leq 2000000$
1	x y A	Add A to the number of customers in table square (x, y). A will be a positive integer, not exceeding 10000.
2	X1 Y1 X2 Y2	Query the current number of customers in squares (x, y), where $X1 \leq x \leq X2$ and $Y1 \leq y \leq Y2$.
3	None	Terminate program. This instruction is given only once and it will be the last instruction

The number of "1" (Add) instructions will not exceed 160 000. The number of "2" (Query) instructions will not exceed 10000.

Output

Your program must not answer anything to input lines with an instruction other than 2. If the instruction is 2, then your program is expected to answer the query by writing the answer as a single line containing a single integer to the output file.

Example

mokia.in	mokia.out

0 4
1 2 3 3
2 1 1 3 3
1 2 2 2
2 2 2 3 4
3
	3
	5

Problem I. Stairway

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

Saharna is a beautiful, sceneric place in Moldova where, among all the caves and waterfalls, one can find a variety of stones of different forms and sizes. In the Middle Ages, these stones were used to construct the stairways of the fortresses. As a rule, each step of the stairway was formed of just one stone.

The stones are heavy and for this reason their positions are fixed along a string, that is the stones are in a given order. The craftsmen know the heights of the stones, hence they have a sequence of integers $H = (h1, \ldots, h_n)$, where h_i represents the height of the i-th stone.

To build a stairway, the craftsmen went along the string and selected consecutively a stone for each step of the stairway in construction. Obviously, a stone can be selected only when its height is not lower than the height of the previously selected stone.

As an example, if $H = (1, 3, 4, 2, 3, 4, 1, 2, 2, 3, 3, 2)$, to build a stairway one can select the underlined stones in the sequence below: $H = (\underline{1}, 3, 4, \underline{2}, 3, 4, 1, \underline{2}, \underline{2}, \underline{3}, \underline{3}, 2)$.

Since bigger is necessarily better, to build a better castle the craftsmen had to use as many stones as possible for the construction of the stairways.

We denote by $L(H, k)$ the maximal number of stones that can be used to build k stairways, each having at least one step. For the above example, it is not hard to see that $L(H, 1) = 6$, that is, the underlined stones represent an optimal stairway.

Similarly, one can check that $L(H, 2) = 9$, $L(H, 3) = 12$, etc,

It is also clear that by taking k to be consecutively 1, 2, 3, and so on, at some point, for some number q, $L(H, q) = n$, where n is the total number of stones.

Your task is to help the medieval craftsmen and write a program which, given the sequence of heights H, computes the maximal number of stones that can be used in k stairways — $L(H, k)$, where $k = 1, 2, \ldots, q$.

Input

The input file contains on the first line the positive integer n. The second line of the file contains the positive integers $h_1, h_2, \ldots, h_n$, separated by blanks. $1 \leq n \leq 5000$, $1 \leq h_i \leq 255$

Output

The output file must contain on each of the q lines a positive integer. The k-th line of the file will contain the number $L(H, k)$.

Example

stairway.in	stairway.out
12 1 3 4 2 3 4 1 2 2 3 3 2	6 9 12

Problem J. Toponyms

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

A toponym is the name given to a city, village, river, mountain etc. Very often, in the Republic of Moldova, one can find toponyms which are very similar. For example, Orhei and Orheiul Vechi; Jora de Sus, Jora de Mijloc and Jora de Jos.

As a rule, every toponym represents a sequence consisting of the characters A, B, C, ..., Z, a, b, c, ..., z and blank character. In toponyms there can not appear sequences of two or more consecutive blanks. Toponyms have no leading or trailing blanks. The subsequences consisting of the first m characters of the toponym is called a prefix of length m. For example, the subsequence Jora, is a prefix of length 4 of the toponym Jora de Mijloc.

Level of similarity $Ls(T)$ of a set T of toponyms is defined as the length of the longest common prefix of the toponyms from T. For example, for the set of toponyms T = Jora de Sus, Jora de Mijloc, Jora de Jos, the level of similarity $Ls(T) = 8$.

Level of complexity $Lc(T)$ of a set T of toponyms is defined as $Ls(T) \times k$ where k is the number of toponyms in T.

For example, for the set of toponyms T = Jora de Sus, Jora de Mijloc, Jora de Jos, the level of complexity $Lc(T) = 24$.

Write a program which, for a given set of toponyms S, find the subset T, $T \subset S$, with the maximal level of complexity.

Input

The input file contains on the first line an integer n — number of toponyms in S. Each of the next n lines of the input file contains a toponym. Each toponym is a string of characters A, B, C, …, Z, a, b, c, …„ z and the blank.

Output

The output file must contain a single line with an integer representing the maximal level of complexity $Lc(T)$.

Example

toponyms.in	toponyms.out
7 Jora de Sus Orhei Jora de Mijloc Joreni Jora de Jos Japca Orheiul Vechi	24