A. Dynamic PQ Trees

PQ trees are data structures for representing complex permutations. It has many applications in graph theory. Besides leaves, there are two kinds of internal nodes in PQ trees: sets and sequences. For any set, reordering it results in an equivalent set, like {`3,7,5,2`} and {`7,2,5,3`}. On the other hand, a sequence can only be reversed but cannot be shuffled in any other ways. `<6,9,4,1>` is equivalent to `<1,4,9,6>`, but not `<4,9,1,6>`. Both sets and sequences can be contained as an element in other sets or sequences. The only constraint is that any two leaves of a PQ tree must have different values.

It is interesting to count how many equivalent PQ trees of a PQ tree. But before that, we have to build the PQ trees! We would like to build the trees dynamically, using the PQ language described below. You are requested to write an interpreter for this language, which will build the trees and print the count when asked.

There are 26 variables in PQ language, denoted by lower-case letters, which are initially $NULL$. We can assign to them a set, a sequence or a non-negative integer (leaf). These variables can then become an element of other sets or sequences, as being sub-trees of other PQ trees. Also, we would like to support insertion and deletion for existing sets and sequences. Another feature of PQ language, which is the most important one, is to count the number of equivalent PQ trees. Following is the formal grammar of this language. The symbol ␣ denotes a white space character.

$$
\begin{aligned}
Command &:= Assignment \mid Operation \\
Assignment &:= Reference \texttt{ = } Value \\
Operation &:= Insertion \mid Deletion \mid Counting \\
Insertion &:= \texttt{INSERT}\sqcup Reference \sqcup Value \\
Deletion &:= \texttt{DELETE}\sqcup Reference \sqcup Value \\
Counting &:= \texttt{COUNT}\sqcup Reference \\
Reference &:= Variable \mid Reference \texttt{ [} Expression \texttt{]} \\
Value &:= Set \mid Sequence \mid Expression \mid Reference \\
Set &:= \{\ ValueList\ \} \\
Sequence &:= \texttt{<}\ ValueList\ \texttt{>} \\
ValueList &:= Value \mid Value\ \texttt{,}\ ValueList \\
Expression &:= Term \mid Expression \texttt{ + } Term \mid Expression \texttt{ - } Term \\
Term &:= Factor \mid Term \texttt{ * } Factor \mid Term \texttt{ / } Factor \\
Factor &:= IntValue \mid IntValue \texttt{ ^ } IntValue \\
IntValue &:= Constant \mid Reference \mid \texttt{(}\ Expression\ \texttt{)} \\
Variable &:= \texttt{a} \mid \ldots \mid \texttt{z} \\
Constant &:= \texttt{0} \mid \texttt{1} \mid \texttt{2} \mid \ldots
\end{aligned}
$$

- For *Insertion* and *Deletion*, the *Reference* must be either a set or a sequence.

- For *Deletion*, every elements with value equivalent to *Value* is removed. This operation causes no effect if no such element is found in the *Reference*, or this element is the only one in the *Reference*.
- For *Counting*, the number of equivalent PQ trees of the *Reference* should be outputted. Since the number may be large, please output the value modulo 1000003.
- Note that the assignments in PQ language is by reference for sets and sequences, and by value for integers. For example, the following program would assign both variables to the same sequence, `<1,2>`.

```
a = <1>
b = a
INSERT b a[0]+1
```

- For *Reference*, the square brackets can only be used on sequences. It refers to the n-th element in this sequence, where n is the result of the *Expression* inside the square bracket. Note that the index starts from 0.
- For *Term*, the remainder of division, if any, is dropped. The operators `*` and `/` are applied from left to right, i.e., `a/b*c` means `(a/b)*c`. The same order is applied on `+` and `-`.
- The *Constant* can be any non-negative integer less than 2^{31}.
- A "`SyntaxError`" is raised if the command does not follow the grammar.
- A "`NullPointerError`" is raised when the command requests value of a *NULL* variable, or the *Reference* of insertion/deletion/counting is *NULL*.
- A "`TypeError`" is raised when the command encounters a type mismatch. For example, inserting into a variable which holds an integer, doing arithmetic operations on a sequence, using square brackets on a set, or mapping *IntValue* to a non-integer value.
- An "`OverflowError`" is raised when any intermediate calculation results in a number larger than $2^{31} - 1$ or less than 0, or a *Constant* is not in this range, or any number is divided by zero.
- An "`OutOfRangeError`" is raised when the result of *Expression* inside a square bracket is larger than or equal to the size of the referring sequence.
- A "`DuplicatedLeavesError`" is raised when the command causes any internal nodes have two or more leaves with the same value.
- Whenever encountering an error, that command will cause no effect and "`Error`" is outputted. If two or more errors occur, only one "`Error`" should be outputted.
- If the command is not *Counting* and does not raise any error, output "`.`"

Input Format

The input contains a program written in PQ language. Each line of the program contains one command. There are at most 250 lines in the input, and each line contains at most 100 characters.

Output Format

Please output one line for each command, following the specification above.

Sample Input

```
a={1}
b=a
INSERT a 2
COUNT a
COUNT b
b=a[0]
a={1,2,5}
b=<2,4,a>
b=<3,4,a>
COUNT b
DELETE a 2
COUNT b
COUNT c
c=2^31-1
c=3+5/2+(8*7*6)^2
d=<9>
INSERT d c
INSERT d d[100]
INSERT d a
COUNT c
COUNT d
c={1,5}
d={c,a}
COUNT d
d[1]={d[0]/3,d[0]+b[1]}
INSERT d[1] d
NO_THIS_COMMAND
```

Sample Output

```
.
.
.
2
2
Error
.
Error
.
12
.
4
Error
Error
.
.
.
Error
.
1
4
.
Error
4
.
Error
Error
```

(Almost empty page.)

B. Recurrence

Consider a tuple $P_1, P_2, \ldots, P_n$. Now consider the following recurrence function.

- $F(P_1, P_2, \ldots, P_n) = 0$, if any of the P_i is negative or the tuple P is not sorted in non-increasing order.
- $F(P_1, P_2, \ldots, P_n) = 1$, if all of the P_i's is zero.
- $F(P_1, P_2, \ldots, P_n) = F(P_1 - 1, P_2, P_3, \ldots, P_n) + F(P_1, P_2 - 1, P_3, \ldots, P_n) + F(P_1, P_2, P_3 - 1, \ldots, P_n) + \cdots + F(P_1, P_2, \ldots, P_n - 1)$ otherwise.

For example if n is 4 then the value:
$F(4, 3, 2, -1)$ is 0 because the last parameter is negative.
$F(4, 3, 2, 5)$ is 0 because the tuple is not sorted from the largest to smallest.
$F(4, 3, 2, 1) = F(3, 3, 2, 1) + F(4, 2, 2, 1) + F(4, 3, 1, 1) + F(4, 3, 2, 0)$.
$F(1, 1, 0, 0) = F(0, 1, 0, 0) + F(1, 0, 0, 0) + F(1, 1, -1, 0) + F(1, 1, 0, -1) = 1$

Given the tuple P your task is to calculate the value of $F(P_1, P_2, P_3, \ldots, P_n)$. The result can be very big so output the result mod 1000000009 (this is a prime number).

Input

There are multiple test cases in the input file, terminated by EOF.

Each test case consists of two lines. First line contains n. Second line contains n integers separated by a single space. These are the tuple P. n is between 1 and 1000 inclusive. Each of the numbers in tuple P is between 1 and 1000 inclusive. P will be sorted in non-increasing order.

Output

For each test case output contains a line containing the value of $F(P_1, P_2, P_3, \ldots, P_n)$ mod 1000000009.

Sample Input

```
3
7 5 4
6
7 7 5 3 2 1
2
4 2
3
7 4 4
4
8 7 5 5
5
7 7 6 5 5
2
8 7
3
6 3 1
```

Sample Output

```
100100
398009117
9
25025
923714728
311516464
1430
315
```

C. Scott's New Trick

Little Scott recently learned how to perform arithmetic operations modulo some prime number P. As a training set he picked two sequences **a** of length N and **b** of length M, generated in the following way:

- $a_1 = A_1$
- $a_2 = A_2$
- $a_i = (a_{i-2} \times A_3 + a_{i-1} \times A_4 + A_5) \bmod P$, for $i = 3 \ldots N$
- $b_1 = B_1$
- $b_2 = B_2$
- $b_j = (b_{j-2} \times B_3 + b_{j-1} \times B_4 + B_5) \bmod P$, for $j = 3 \ldots M$

Now he wants to find the number of pairs (i, j), where $1 \le i \le N$ and $1 \le j \le M$, such that $(a_i \times b_j) \bmod P < L$, for given number L. He asked you to do the same to help him check his answers.

Input

The first line of input file consists of a single number T, the number of test cases. Each test consists of three lines. The first line of a test case contains two integers: prime number P and positive integer L. The second line consists of six non-negative integers $N, A_1, A_2, A_3, A_4, A_5$. Likewise, the third line contains six non-negative integers $M, B_1, B_2, B_3, B_4, B_5$.

Output

Output T lines, with the answer to each test case on a single line.

Constraint

- $T = 20$
- $2 \le P < 250000$
- P is prime
- $1 \le L \le P$
- $2 \le N, M \le 10000000$
- $0 \le A_1, A_2, A_3, A_4, A_5, B_1, B_2, B_3, B_4, B_5 < P$

Sample Input

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

Sample Output

```
6
10
15
3
9
```

D. Geometric Map

Your task in this problem is to create a program that finds the shortest path between two given locations on a given street map, which is represented as a collection of line segments on a plane.

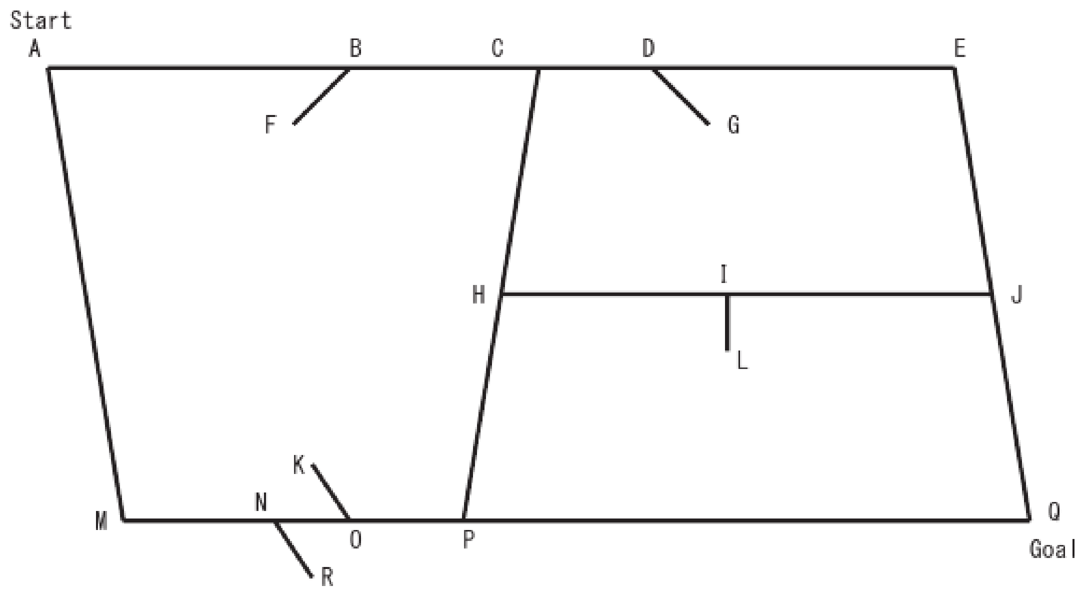

Figure 4: An example map

Figure 4 is an example of a street map, where some line segments represent streets and the others are signs indicating the directions in which cars cannot move. More concretely, AE, AM, MQ, EQ, CP and HJ represent the streets and the others are signs in this map. In general, an end point of a sign touches one and only one line segment representing a street and the other end point is open. Each end point of every street touches one or more streets, but no signs.

The sign BF, for instance, indicates that at B cars may move left to right but may not in the reverse direction. In general, cars may not move from the obtuse angle side to the acute angle side at a point where a sign touches a street (note that the angle CBF is obtuse and the angle ABF is acute). Cars may directly move neither from P to M nor from M to P since cars moving left to right may not go through N and those moving right to left may not go through O. In a special case where the angle between a sign and a street is rectangular, cars may not move in either directions at the point. For instance, cars may directly move neither from H to J nor from J to H.

You should write a program that finds the shortest path obeying these traffic rules. The length of a line segment between (x_1, y_1) and (x_2, y_2) is $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

Input

The input consists of multiple datasets, each in the following format.

```
n
xs ys
xg yg
x11 y11 x12 y12
...
xk1 yk1 xk2 yk2
...
xn1 yn1 xn2 yn2
```

n, representing the number of line segments, is a positive integer less than or equal to 200.

(x_s, y_s) and (x_g, y_g) are the start and goal points, respectively. You can assume that $(x_s, y_s) \neq (x_g, y_g)$ and that each of them is located on an end point of some line segment representing a street. You can also assume that the shortest path from (x_s, y_s) to (x_g, y_g) is unique.

(x_1^k, y_1^k) and (x_2^k, y_2^k) are the two end points of the k th line segment. You can assume that $(x_1^k, y_1^k) \neq (x_2^k, y_2^k)$. Two line segments never cross nor overlap. That is, if they share a point, it is always one of their end points.

All the coordinates are non-negative integers less than or equal to 1000. The end of the input is indicated by a line containing a single zero.

Output

For each input dataset, print every street intersection point on the shortest path from the start point to the goal point, one in an output line in this order, and a zero in a line following those points. Note that a street intersection point is a point where at least two line segments representing streets meet. An output line for a street intersection point should contain its x- and y-coordinates separated by a space.

Print '`-1`' if there are no paths from the start point to the goal point.

Sample Input

```
8
1 1
4 4
1 1 4 1
1 1 1 4
3 1 3 4
4 3 5 3
2 4 3 5
4 1 4 4
3 3 2 2
1 4 4 4
9
1 5
5 1
5 4 5 1
1 5 1 1
1 5 5 1
2 3 2 4
5 4 1 5
3 2 2 1
4 2 4 1
1 1 5 1
5 3 4 3
11
5 5
1 0
3 1 5 1
4 3 4 2
3 1 5 5
2 3 2 2
1 0 1 2
1 2 3 4
3 4 5 5
1 0 5 2
4 0 4 1
5 5 5 1
2 3 2 4
0
```

Sample Output

```
1 1
3 1
3 4
4 4
0
-1
5 5
5 2
3 1
1 0
0
```

(Almost empty page.)

E. Godzilla

Godzilla terrorizes Byteland again. Every day the monster comes out from the ocean, it moves along the streets and destroy a building with eating all people in there. Godzilla moves so slow such that every day it can only choose (ans must choose) a new building to attack. Godzilla will eat all people who are still lies in it. Every night Godzilla must go back into the ocean. People are terrified so that every night there is exactly one person in each building (if there is any) flees to the countryside.

In Byteland, every building is built on the intersections. At every intersections there is exactly one building. Intersections are connected by two-way streets. If Godzilla chose to destroy one building at an intersection. It will destroy everything nearby including breaking all the streets that lead to the intersection.

What is the maximum number of people Godzilla can eat?

Input

There are multiple test cases in the input file, terminated by EOF. For each test case:

The first line contains n and m ($1 \leq n \leq 100000, 1 \leq m \leq 500000$) denoting the number of intersections and streets. Intersections are numbered from 1 to n, where intersection 1 is the only intersection located on the shores of the ocean. The next row contains $k_1, k_2, \ldots, k_n$ denoting number of people live in each building. ($0 \leq k_i \leq 100000$). In each of the next m lines there are two integers a_i and b_i ($1 \leq a_i, b_i \leq n$, $a_i \neq b_i$), which means there is a street connects intersection a_i and b_i. Initially every intersection can be reached from intersection 1.

Output

Output the maximum number of people that Godzilla can eat.

Sample Input

```
5 5
1 3 2 4 7
1 2
1 3
2 3
2 4
3 5
3 3
1 2 3
1 2
2 3
1 3
```

Sample Output

```
11
3
```

(Almost empty page.)

F. Checker

Tmt loves to play math on chessboard. One day he takes out a chessboard of $n \times n$ and exactly n rooks numbered from 1 to n. He wishes to place all the rooks on the board such that no two rooks can attack each other. Moreover, for each i, the i-th rook, would not be placed on the i-th row or i-th column. Tmt wants to know how many ways are there to place all the rooks on the board. The number is too large so he ask your help for telling him the answer modulo m.

Input

There are multiple test cases in the input file, terminated by EOF. For each test case, there is only two integers n and m. ($1 \le n \le 10^{18}, 1 \le m \le 10^6$.)

Output

For each test case please output the answer on a single line.

Sample Input

```
3 120
```

Sample Output

```
4
```

(Almost empty page.)

G. The Teacher's Side of Math

One of the tasks students routinely carry out in their mathematics classes is to solve a polynomial equation. It is, given a polynomial, say $x^2 - 4x + 1$, to find its roots $(2 \pm \sqrt{3})$.

If the students' task is to find the roots of a given polynomial, the teacher's task is then to find a polynomial that has a given root. Ms. Galsone is an enthusiastic mathematics teacher who is bored with finding solutions of quadratic equations that are as simple as $a + b\sqrt{c}$. She wanted to make higher-degree equations whose solutions are a little more complicated. As usual in problems in mathematics classes, she wants to maintain all coefficients to be integers and keep the degree of the polynomial as small as possible (provided it has the specified root). Please help her by writing a program that carries out the task of the teacher's side.

You are given a number t of the form:

$$t = \sqrt[m]{a} + \sqrt[n]{b}$$

where a and b are distinct prime numbers, and m and n are integers greater than 1.

In this problem, you are asked to find t's minimal polynomial on integers, which is the polynomial $f(x) = a_d x^d + a_{d-1}x^{d-1} + \ldots + a_1 x + a_0$ satisfying the following conditions.

- Coefficients $a_0, \ldots, a_d$ are integers and $a_d > 0$.
- $f(t) = 0$.
- The degree d is minimum among polynomials satisfying the above two conditions.
- $f(x)$ is primitive. That is, coefficients $a_0, \ldots, a_d$ have no common divisors greater than one.

For example, the minimal polynomial of $\sqrt{3} + \sqrt{2}$ on integers is $f(x) = x^4 - 10x^2 + 1$. Verifying $f(t) = 0$ is as simple as the following ($\alpha = \sqrt{3}, \beta = \sqrt{2}$).

$$\begin{aligned}
f(t) &= (\alpha + \beta)^4 - 10(\alpha + \beta)^2 + 1 \\
&= (\alpha^4 + 4\alpha^3\beta + 6\alpha^2\beta^2 + 4\alpha\beta^3 + \beta^4) - 10(\alpha^2 + 2\alpha\beta + \beta^2) + 1 \\
&= 9 + 12\alpha\beta + 36 + 8\alpha\beta + 4 - 10(3 + 2\alpha\beta + 2) + 1 \\
&= (9 + 36 + 4 - 50 + 1) + (12 + 8 - 20)\alpha\beta \\
&= 0
\end{aligned}$$

Verifying that the degree of $f(t)$ is in fact minimum is a bit more difficult. Fortunately, under the condition given in this problem, which is that a and b are distinct prime numbers and m and n greater than one, the degree of the minimal polynomial is always mn. Moreover, it is always monic. That is, the coefficient of its highest-order term (a_d) is one.

Input

The input consists of multiple datasets, each in the following format.

```
a m b n
```

This line represents $\sqrt[m]{a} + \sqrt[n]{b}$. The last dataset is followed by a single line consisting of four zeros. Numbers in a single line are separated by a single space.

Every dataset satisfies the following conditions.

1. $\sqrt[m]{a} + \sqrt[n]{b} \leq 4$.
2. $mn \leq 20$.
3. The coefficients of the answer $a_0, \ldots, a_d$ are between $(-2^{31}+1)$ and $(2^{31}-1)$, inclusive.

Output

For each dataset, output the coefficients of its minimal polynomial on integers $f(x) = a_d x^d + a_{d-1}x^{d-1} + \ldots + a_1 x + a_0$, in the following format.

```
ad  ad-1 ... a1  a0
```

Non-negative integers must be printed without a sign (`+` or `-`). Numbers in a single line must be separated by a single space and no other characters or extra spaces may appear in the output.

Sample Input

```
3 2 2 2
3 2 2 3
2 2 3 4
31 4 2 3
3 2 2 7
0 0 0 0
```

Sample Output

```
1 0 -10 0 1
1 0 -9 -4 27 -36 -23
1 0 -8 0 18 0 -104 0 1
1 0 0 -8 -93 0 24 -2976 2883 -32 -3720 -23064 -29775
1 0 -21 0 189 0 -945 -4 2835 -252 -5103 -1260 5103 -756 -2183
```