Problem A. Colonies

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

Little Anya studies the life of bacteria. For the next experiment, she took the liquid containing some interesting bacteria, poured it on a square glass, and then photographed the glass with an electronic microscope.

The microscope saves the photos in compressed format and records each photo as a sequence of numbers 0, 1, and 2. When the microscope has to record a photo of a certain square, which initially is the square corresponding to the whole photo, it acts according to the following algorithm. If there are no bacteria in the whole square, it is denoted by a single number 0. If, conversely, the whole square is filled with bacteria, this is denoted by a single number 1. In the other cases, the record starts with the number 2, after which the microscope records the contents of four lesser squares: first the top left quarter of the initial square, then the top right quarter, then the bottom right quarter, and finally the bottom left quarter. The microscope uses the same algorithm to record each of the four lesser squares.

Anya got the recorded photo, and now wants to find out how many colonies of bacteria there were on the glass. Two squares on the photo which are filled with bacteria belong to the same colony if a bacterium can travel from one of them to the other by moving between neighboring squares filled with bacteria. Two squares filled with bacteria are neighboring if there exists a segment of strictly positive length which belongs to the border of both squares.

Solve the generalized version of Anya's problem: given a recorded photo, find out how many colonies of bacteria are represented on the photo.

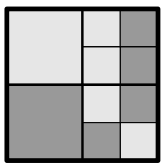
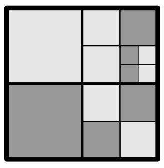
Input

The first line contains the photo of the square glass in compressed format. The line consists of numbers 0, 1, and 2 separated by spaces, and is constructed according to the algorithm described in the statement. The line contains between 1 and 15 001 numbers, inclusive.

Output

Print one integer: the number of colonies of bacteria.

Examples

standard input	standard output	Notes
`2 0 2 0 1 1 0 2 0 1 0 1 1`	`2`	
`2 0 2 0 1 2 1 0 0 1 0 2 0 1 0 1 1`	`2`	

Problem B. Cube Puzzle

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

Alice got a puzzle as her March 8 present. The puzzle consists of six parts which represent the faces of a cube, possibly with ragged edges. When assembled, the parts constitute the border of an $n \times n \times n$ cube. The interior of the cube is empty.

To formally define the puzzle, let us divide the space into three-dimensional $1 \times 1 \times 1$ cells and introduce a coordinate system. The cube consists of cells with coordinates (x, y, z) where $1 \le x, y, z \le n$. The cells with $2 \le x, y, z \le n - 1$ are inner cells, and the remaining cells of the cube are border cells.

When the cube is assembled, each part lies completely in the cube and does not intersect other parts, each border cell is fully covered by one of the parts, and inner cells have no common volume with the parts. Furthermore, if two border cells lie on the same face of the cube and do not belong to other faces, they must be covered by the same part.

All parts are flat: all cells covered by the same part will have one of the coordinates equal among them. All parts are connected: if we put any part on a flat surface, the resulting plane figure will be connected by side. When assembling the cube, one can rotate, flip, and translate the parts.

Solve the generalized version of Alice's puzzle. Given the descriptions of the six parts, output what the assembled cube looks like.

Input

The first line contains an integer n ($4 \le n \le 10$). Then follow descriptions of the six parts. Each description consists of n lines with n characters each. There, j-th character of i-th line describes the cell (i, j): it is either "`.`" if the cell is empty or an uppercase English letter from "`A`" to "`F`" if it is covered by the part. In the first part's description, the letter is always "`A`", in the second one "`B`", ..., and in the sixth one, it is "`F`". The part descriptions are separated by a single empty line.

Output

Print n blocks, each block containing n lines with n characters each. There, k-thcharacter of j-th line of i-th block must describe the cell with coordinates (i, j, k): it must be either "`.`" if the cell is empty or an uppercase English letter from "`A`" to "`F`" if it is covered by the respective part. The blocks must be separated by a single empty line.

If there are multiple possible solutions, print any one of them. It is guaranteed that, in all tests prepared by the jury, the solution exists.

Example

standard input	standard output
4	BEBB
..A.	BBBF
AAAA	ABBB
AAA.	DDDD
..A.	
	AEEE
BBB.	A..F
.BBB	A..F
BB.B	ADDF
....	
	AEEF
.C..	A..F
CCCC	A..F
.CC.	DDDD
.CCC	
	EECF
.D.D	CCCC
.DDD	ACCF
.DDD	CCCD
DD.D	
..E.	
.EE.	
EEEE	
E...	
..FF	
FFF.	
.FFF	
.F..	

Problem C. Electricity

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

Vasya crafts an electric circuit. He has four pins on a board, located in corners of a square. He wants to connect two pairs of pins using straight segments of foil: A_1 with A_2, and B_1 with B_2. Obviously it is no good if these two segments have at least one common point because that may cause a short circuit. Check whether the short circuit may happen.

Input

The first line contains a single integer T ($1 \le T \le 64$). After that, T test cases follow, each test case consists of two lines.

The first line of each test case contains four integers: coordinates of A_1 (x, y) followed by coordinates of A_2. The second line of each test case contains coordinates of B_1 and B_2 in the same format.

All coordinates are between 0 and 1, inclusive.

Output

For each test case, print "`YES`" (quotes for clarity) on a separate line if two segments have at least one common point and "`NO`" (quotes for clarity) otherwise.

Examples

standard input	standard output

1
0 0 1 0
0 1 1 1
	NO

2
0 0 1 1
1 0 0 1
0 0 0 0
1 1 1 1
	YES
	NO

Problem D. Comparatively Honest Healing

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

Little Andrey plays a computer game. Andrey the Chaos Knight is his character in the game. Currently Andrey fights against an archangel of Order to continue his glorious journey to world domination.

A recent strike of archangel's sword lowered Andrey's health to a dangerous level. Unfortunately, due to Chaos influence, the healing potions which Andrey prepared for the fight transformed into pumpkin juice, but he absolutely has to find some way of healing: in this game, death is irreversible. Fortunately, Andrey knows approximately four hundred comparatively honest methods to heal his hero, so perhaps the Chaos Knight will survive the fight.

One of the ways to heal is related to how the character's health changes when he puts some equipment on or takes it off. In each moment, the character's health is characterized by two integers: the current number of health points H and the maximum number of health points L. As soon as the number H becomes non-positive, the character dies.

Some wearable items increase the health point limit when equipped. There is a non-negative number corresponding to each wearable item: how much the value L increases if this item is equipped.

The items can be put on or taken off, and, importantly for our hero, such actions do not take game time. If the hero puts some wearable item on or takes it off, both L and H are recalculated. Specifically, if the old upper limit was L_1 and the new one is L_2, then the new number of health points H_2 is proportionally recalculated from the old number H_1 by the following formula:

$$H_2 = H_1 \cdot L_2 / L_1.$$

The calculation uses real numbers, and after that, H_2 is rounded to the nearest integer. The numbers of form $z + 0.5$ for integer z are rounded to the nearest even number: 0.5 is rounded to 0, 1.5 and 2.5 to 2, 3.5 and 4.5 to 4, and so on.

If multiple items are being put on or taken off sequentially, recalculation and rounding take place immediately after each such action.

The Chaos Knight has several wearable items which impact the maximum number of health points L. Andrey does not like to carry around more than is necessary, so all these items have different types: each of them can be put on or taken off independently of the others. Some of these items can be already equipped.

Currently, Andrey has S health points, and the upper limit is L. To fight comfortably, he would like to have at least T health points. Additionally, Andrey wants the Knight to be equipped with exactly the same set of items which influence his health as before the re-dressing. The thing is, when solving this problem, Andrei does not consider items which do not influence the health limit. Actually, it may happen that such items should be taken off before the process to free some item slots, and after the process, they have to be put on again for their other qualities useful in a fight.

Solve the generalized version of Andrey's problem: given integers S, T, and L, as well as the list of wearable items which influence the maximum number of health points, find a sequence of actions after which the character will wear the same equipment as before but have at least T health points, or determine that the described method of healing will not work this time.

Input

The first line contains four integers: the number of wearable items n, the initial number of health points S, the desired number of health points T and the initial maximum number of health points L ($1 \le n \le 10$, $1 \le S \le T \le L \le 1000$). Each of the next n lines describes a single wearable item. The description has the form “c x *name*”. Here, c is either “+” if the item is already equipped or “-” if not, x is a positive integer which is added to the health limit when the item is equipped, and *name* is the name of this item. The name consists of one or more words separated by spaces, and each word consists of lowercase English letters. The length of a name is from 1 to 20 characters. A name can not start or end with a space and can not contain two consecutive spaces. All given items have different names.

The value L already takes all equipped items into account. It is guaranteed that if all items are taken off, the health limit will be at least one, and if they are all put on, it will be at most one thousand.

Output

If healing the hero to T or more health points is possible with the described method, print a sequence of actions. On the first line, print k, the number of actions. After that, print $2k + 1$ lines. First, print the state before all actions, then the first action, then the state after the first action, then the second action, and so on. Each state must be printed in the form “H / L”: the number of health points, a slash, and the health limit, separated by spaces. Each action must be printed in the form “c *name*”. Here, c is either “+” if the action is to put the item on and “-” if the action is to take it off, and *name* is the name of the item. It is not allowed to put on an equipped item or take off an unequipped one.

If there are several possible sequences of actions, print any one of them. You do not have to minimize k, however, it must be at most two million. It is guaranteed that, if there exists a solution, there also exists a solution satisfying this restriction.

In case the described method does not allow to achieve the goal, print -1 instead of k on the first line.

Examples

standard input	standard output

`1 1 2 3`
`+ 1 warm clothes`
	`2`
	`1 / 3`
	`- warm clothes`
	`1 / 2`
	`+ warm clothes`
	`2 / 3`

`1 1 2 2`
`+ 1 warm clothes`
	`-1`

`3 23 30 50`
`+ 14 ring of the titans`
`- 1 amulet of health`
`+ 29 demon shield`
	`8`
	`23 / 50`
	`+ amulet of health`
	`23 / 51`
	`- ring of the titans`
	`17 / 37`
	`- demon shield`
	`4 / 8`
	`- amulet of health`
	`4 / 7`
	`+ amulet of health`
	`5 / 8`
	`+ ring of the titans`
	`14 / 22`
	`- amulet of health`
	`13 / 21`
	`+ demon shield`
	`31 / 50`

Problem E. Matrix Power

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

You are given a square matrix A consisting of integers. Find the least positive integer k such that the equation $A^k = 0 \pmod{10^9+7}$ is satisfied. In other words, every element of the matrix A^k must be evenly divisible by $10^9 + 7$.

Input

The first line contains an integer n ($1 \le n \le 300$).

The next n lines contain the matrix A of size $n \times n$: in the i-th line, the j-th integer is A_{ij} ($0 \le A_{ij} < 10^9 + 7$).

Output

Print k. If there is no such k, print "`Infinity`" (without quotes).

Example

standard input	standard output

3
0 1 0
0 0 1
0 0 0
	3

Problem F. Oddosort

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

In this problem you have to sort a singly-linked linear list on the following conditions:

- You are only allowed to do pairwise comparisons of element values.
- You have only $\mathcal{O}(1)$ extra space.
- The sorting algorithm must be **stable**.

A *singly-linked linear list* is a data structure which holds N values in some order, similar to an array. Values are stored in N *nodes*, each node containing a single value and a pointer to the next node (or a special `nil` node if it is the last node). A list is characterized by a pointer to its *head*: the node containing the first value (in order) or a special `nil` node if the list is empty. Note that nodes themselves are unordered, it's pointers between them that define the order.

A solution for this problem must be a program P in a programming language called Meow++ (described below). The head of the list is provided for your program in a global variable called `head`. The same variable must contain the head of the sorted list when the program terminates. The program you submit to the testing system (denoted P') must print P to the standard output.

Here is an example program in Meow++ which sorts an arbitrary two-element list:

```
a = head
b = head #next
if b #value < a #value {
  head = b
  loop {
    a #next = a #next #next
    break
  }
  b #next = a
} else { }
```

The semantics are similar to that of any modern imperative language like C++, Java, or Python. There are assignments, conditionals and infinite loops whose behavior can be altered with `break` and `continue`.

A program in Meow++ manipulates multiple variables. Each variable can either point to a list node or a special `nil` node. Each node contains two subfields: `#value` and `#next` (pointing to the next node in the list). You can reassign any variable and any `#next` subfield to any pointer. Note that `nil` does not have any subfields, so trying to access or modify them will crash Meow++ program and will result in the "`Wrong Answer`" outcome.

Here is a more formal description of Meow++'s syntax in BNF-like form with some regex pieces:

```
PROGRAM   ::= STATEMENT*  # Asterisk means "zero or more occurrences one after another"
BLOCK     ::= "{" STATEMENT* "}"
STATEMENT ::= POINTER "=" POINTER
            | "if" CONDITION BLOCK "else" BLOCK
            | "loop" BLOCK | "break" | "continue"
CONDITION ::= POINTER "==" POINTER
            | POINTER "#value" "<" POINTER "#value"
POINTER   ::= VARIABLE "#next"{0,10} | "nil"
VARIABLE  ::= [a-zA-Z0-9_]{1,16}  # Between 1 and 16 English letters,
                                  # digits and underscores
```

Note the following subtleties:

- Omitting curly brackets is not permitted.
- Omitting `else` is not permitted.
- There are only two types of conditions: comparing two pointers for equality and comparing two values.
- You cannot build complex conditions as there are no boolean operators.
- All tokens must be whitespace-separated. For example `foo #next` is fine, while `foo#next` is not.
- Name of a variable must not coincide with any of the keywords: `if`, `else`, `loop`, `break`, `continue`, `nil`.
- You cannot write more than ten `#next`s after a variable.

And here are more details about Meow++'s semantics:

- The program is executed step-by-step, starting with the very first statement, and terminating successfully after completing the last statement.
- All global variables except `head` are initialized with `nil`, and `head` is initialized with the input.
- If the program *crashes*, your solution will be judged as "`Wrong Answer`".
- If at any time the program tries to access a subfield of `nil` (either directly or because it calls `foo #bar` where `foo` points to `nil`), the program crashes.
- Assignments of form `nil = B` immediately crash the program.
- Keywords `break` and `continue` affect the **innermost** `loop` only.
- The expression `nil == nil` always evaluates to true, but `nil == foo` may be either true or false. Note that two different nodes may hold the same value.
- Comparison of form `A #value < B #value` compares values which are contained in nodes denoted by `A` and `B`. Both must not evaluate to `nil`, for example, `nil #value < nil #value` crashes the program. It is guaranteed that `<` is a total order on values.

Your program is allowed to execute at most $1000 \cdot (N + 1)$ statements, otherwise your solution will be judged as "`Wrong Answer`". Here the value N is the length of the list. Each iteration of `loop` statement is counted as a single statement.

If your program doesn't correspond to the grammar above or if your program contains more than 10 000 tokens, your solution will be judged as "`Presentation Error`".

Input

The input for P' (the program you submit to the system) contains a single line with text "`Meow++`" (six characters, quotes for clarity).

The input for P (the program in Meow++ printed by P') is given a global variable called `head`. It is guaranteed to contain the head of a correct singly-linked linear list which your program has to sort. The list contains between 0 and 10^5 elements, inclusive.

There are multiple test cases for this problem, each test case is a single fixed list.

Output

P' (the program you submit to the system) must print a correct program P in Meow++.

P must rearrange nodes in the input list so that the list becomes sorted (the order of nodes with the same value must be preserved). P must also ensure that the `head` global variable stores pointer to the first node in the sorted list, or `nil` if it is empty.

Examples

All examples below follow input and output format, but the output programs are not full solutions of the problem.

standard input	standard output	Notes
`Meow++`		The input is 10, 20. This program does nothing in 0 statements.

`Meow++`	`head #next #next = head`
	`head = head #next`
	`head #next #next = nil`
		The input is 20, 10.
		This program swaps the first two elements in 3 statements.

`Meow++`	`last = head`
	`loop {`
	`  if last #next == nil {`
	`    break`
	`  } else { }`
	`  last = last #next`
	`}`
	`last #next = head`
	`head = head #next`
	`last #next #next = nil`
		The input is 30, 10, 20.
		This program moves the first element of the list to the back in 13 statements.

Problem G. Root

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	256 mebibytes

Petya is going to participate in ACM ICPC World Finals this year. That is why he participates in at least two programming contests every day. After these contests, he tries to solve those problems which he couldn't tackle during the contests. Additionally, he studies new algorithms.

Recently, Petya encountered the following problem: solve the equation

$$x^2 \equiv a \pmod{q},$$

where q is a prime number. Petya googled and found an article in Wikipedia with a description of an algorithm which solves this problem in $O(\log q)$.

After that, Petya asked himself: what would happen if he considered equation

$$x^p \equiv a \pmod{q}$$

for any prime number p. Is there any efficient algorithm for solving this more general type of equation? This problem turned out to be too difficult for Petya, so he asks you for help.

Petya understands that such an equation could have a lot of solutions, so he asks you to find at least one of them, or determine that the equation has no solutions.

Input

The first line contains an integer t, the number of test cases ($1 \le t \le 200$).

The following t lines contain test cases. Each line contains three integers p, a, and q. It is guaranteed that p and q are prime numbers and $0 \le a < q$. All numbers in the input do not exceed 10^{18}.

Moreover, it is guaranteed that there are no more than 50 test cases with $p > 10$.

Output

Print exactly t integers, i-th integer must be a solution of the equation

$$x^p \equiv a \pmod{q}$$

given in i-th test case. You must print the integers modulo q.

If the equation has no solutions, print -1.

Example

standard input	standard output

2
2 4 257
2 3 257
	2
	-1

Explanation

In the first test case, $2^2 = 4$.

In the second test case, the equation $x^2 \equiv 3 \pmod{257}$ has no solutions.

Problem H. Smooth Mountains

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

It is not very well-known that there is a large mountain system in northern Peterhof. The thing is that these mountains are very smooth. Consider a set of points $S = \{1, \ldots, n\} \times \{1, \ldots, n\}$: the $n \times n$ square of points with coordinates from 1 to n, inclusive. Denote the height above the sea level at point $(x, y) \in S$ by $h(x, y)$. All point heights are considered nonnegative integers. A mountain system is *m-smooth* on the set S if for any $(x_1, y_1) \in S$ and $(x_2, y_2) \in S$, $|x_1 - x_2| + |y_1 - y_2| \le 1$ implies $|h(x_1, y_1) - h(x_2, y_2)| \le m$. In other words, any two points which are at distance at most 1 have their heights differ by at most m.

You are given integers n, m, k, and $h(x_1, y_1), \ldots, h(x_k, y_k)$ for k distinct points $(x_i, y_i) \in S$. Find the height of every point in S such that the heights of the given k points are as given and the resulting mountain system is m-smooth on the set S, of determine that such mountain system can not exist.

Input

The first line contains three integers: n, m, and k ($1 \le n \le 100$, $1 \le k \le n^2$, $0 \le m \le 10^9$). The next k lines contain three integers each. The i-th of these lines contains x_i, y_i, and $h(x_i, y_i)$ ($1 \le x_i, y_i \le n$, $0 \le h(x_i, y_i) \le 10^9$). It is guaranteed that all the given points (x_i, y_i) are distinct.

Output

If there is an m-smooth mountain system satisfying the given constraints, print "`YES`" on the first line. On each of the next n lines, print n integers. The j-th integer in the i-th of these lines stands for $h(i, j)$. All printed values must be non-negative integers not exceeding 10^9. If there are multiple solutions, print any of them.

If there is no m-smooth mountain system satisfying the constraints, print "`NO`" instead.

Examples

standard input	standard output
`5 4 3`	`YES`
`1 1 0`	`0 2 4 6 8`
`2 2 3`	`2 3 3 5 7`
`5 4 10`	`4 3 5 8 9`
	`6 5 8 10 10`
	`8 8 10 10 10`
`5 10 1`	`YES`
`3 3 0`	`20 15 10 15 20`
	`15 10 5 10 15`
	`10 5 0 5 10`
	`15 10 5 10 15`
	`20 15 10 15 20`
`3 2 2`	`NO`
`1 1 0`	
`3 3 12`	

Problem I. Two Armies

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

The time has come for the final battle between forces of Light and Darkness. The soldiers of the two armies engaged in deadly combat.

There would be no suspense if the forces were not almost equal. Despite the fact that both armies can be vast, it is known that at the start of the final battle, the number of soldiers in the army of Light differs from the number of soldiers in the army of Darkness by at most one thousand.

However, even with this knowledge, there is still no suspense because when Light and Darkness fight, the casualties of both sides are always the same. The final battle will last as long as there is at least one soldier in both armies.

What will be the outcome of the battle?

Input

The first line contains the number of soldiers in the army of Light, and the second one contains the number of soldiers in the army of Darkness. Both numbers are positive integers without leading zeroes, and each number contains at most one million digits.

Output

Print one number: the result of the battle. If X soldiers of Light remain standing, print the number X. If Y soldiers of Darkness emerge victorious, print the negative number $-Y$. Finally, if no soldiers live until the end, print the number 0.

It is guaranteed that, in all tests prepared by the jury, the correct answer will be a number from -1000 to $+1000$, inclusive. You may output the $+$ sign before a non-negative answer.

Examples

standard input	standard output

`1000`
`999`
	`+1`

`12345678900987654123`
`12345678900987654312`
	`-189`