Problem A. Automaton

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	128 Mebibytes

A (non-deterministic) finite automaton is a quintuple $(\Sigma, S, s_0, F, \delta)$, where:

- Σ is the input alphabet (a finite, non-empty set of symbols),
- S is a finite, non-empty set of states,
- $s_0 \in S$ is an initial state,
- $F \subseteq S$ is the set of final states, and
- $\delta\colon S \times \Sigma \to 2^S$ is the transition function.

Note that, unlike in a deterministic finite automaton, the transition function δ returns not a single state, but a set of states. For the purposes of our problem, this set can not be empty.

A finite automaton can *process* a word (a string consisting of symbols of alphabet Σ) as follows:

- Initially, the automaton is in the state $s = s_0$.
- If the word is empty, the processing is finished.
- Otherwise, the first symbol c of the word is consumed, and the current state s is changed to an element of set $\delta(s, c)$; any element may be chosen.

You are given a finite automaton over an alphabet consisting of one symbol. Note that all words in such alphabet are uniquely defined by their length, and the transition function can be thought of as having only one argument (i. e. $\delta\colon S \to 2^S$). We now construct the *partial equivalence* relation in two steps.

On step 1, two states of the automaton are called partially equivalent if there exists such a word that processing of that word can be finished in either of the states.

On step 2, the transitivity rule is applied to the relation obtained on step 1: two states s and t of the automaton are called partially equivalent if there exists a sequence of states r_1, r_2, …, r_k such that $r_1 = s$, $r_k = t$ and each two consecutive states in the sequence are partially equivalent.

Two words are called *almost equivalent* if their processing can be finished in partially equivalent states. Your task is, given two words, to determine if they are almost equivalent.

Input

The first line of input contains two integers: the number of states n and the number of transition descriptions m $(0 < n, m < 200\,000)$. They are followed by m lines describing the transitions. Each of these lines contains two space-separated integers a_i and b_i which mean that a transition from state a_i to state b_i is possible $(1 \le a_i, b_i \le n)$. It is guaranteed that for each state s, there is at least one possible transition from that state (i. e. $\delta(s)$ is not an empty set). The initial state of the automaton has number 1.

The next line contains an integer T—the number of queries $(1 \le T \le 10^5)$. It is followed by T lines containing integers s_i and l_i—the lengths of the words in query i $(0 \le s_i, l_i \le 10^9)$.

Output

Output T lines with answers to the queries. Print "`Yes`" in the i-th line if the words in query i are almost equivalent and "`No`" otherwise.

Example

standard input	standard output

1 1
1 1
1
1 2
	Yes

Problem B. Bijections

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 second
Memory limit:	128 Mebibytes

Consider all bijections of the set $\{1, 2, \ldots, n\}$, that is, mappings f from that set to itself such that $f(1)$, $f(2)$, ..., $f(n)$ is a permutation of 1, 2, ..., n. Number b is called *reachable* from number a if b occurs in the sequence $f(a)$, $f(f(a))$, $f(f(f(a)))$, ... at least once.

Given an integer k, find out how many bijections of this set exist such that the numbers 2, 3, ..., k are reachable from 1. Because this number can be very large, you are asked to represent it in base p and output the last non-zero digit of the result.

Input

The first line of input contains three integers n, k and p ($0 < k \leq n < 2^{10\,000}$, $1 < p < 10^6$ is prime).

Output

The first and only line of output should contain the last non-zero digit of the number of such bijections in base p. The digit itself should be expressed in decimal notation.

Example

standard input	standard output
`2 1 2`	`1`

Explanation

In the above example, $k = 1$, so there are no restrictions on the bijections. The total number of bijections of the set $\{1, 2\}$ is $2_{10} = 10_2$; the last non-zero digit of the answer in base $p = 2$ is therefore 1.

Problem C. Egyptian Tale

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 second
Memory limit:	128 Mebibytes

Vasya is fond of ancient Egyptian culture. He read in the Internet that two things of the most interest to the ancient Egyptians were mathematics and pyramids. It's the same website where he had found out that his favourite scientist Leonardo da Pisa studied in Egypt in his youth. "What a coincidence!" Vasya exclaimed.

Vasya spent some more time in the Internet reading various mathematical articles. It's no surprise he fell asleep on his keyboard, and this always leads to unpleasant consequences. In this particular case Vasya had a dream in which he was sitting on a seashore in a white turban, blood-lusting sharks peeked out of the sea, a laptop was in front of Vasya and a weird task was due to be solved.

"Well, that was a strange dream," Vasya thought, "though maybe…" and he sat down in front of his keyboard again.

The task was to calculate the "pyramidal" Fibonacci number F_N modulo m where the "pyramid" is

$$N = a_1^{a_2^{a_3^{\cdot^{\cdot^{\cdot^{a_n}}}}}}.$$

Recall that Fibonacci numbers are defined as $F_0 = 0$, $F_1 = 1$, $F_{n+2} = F_n + F_{n+1}$ for $n > 1$.

Input

The number of test cases T is given on the first line of input ($0 < T \le 500$). Each test case consists of two lines. The first of these lines starts with the height of the pyramid n followed by n numbers a_i that form the pyramid ($0 < n \le 10$, $0 < a_i \le 10^9$). The number m ($0 < m < 2^{32}$) is given on the second line.

Output

For each test case, output F_N modulo m on a separate line.

Example

standard input	standard output

2
2 2 2
13
2 2 3
13
	3
	8

Problem D. Cut the String

Input file:	`standard input`
Output file:	`standard output`
Time limit:	7 seconds
Memory limit:	128 Mebibytes

Lowercase English letters are written on a cyclic ribbon. At least k equal words must be cut out of it in order to play a role-playing game. Longer words make the game more interesting. That's why your task is to calculate the the length of the longest word that can be cut out at least k times.

Input

Input file consists of several test cases, one test case per line: an integer number k and a non-empty string that represents the letters written on the ribbon. The input is terminated by the end of file. The total number of letters in all testcases doesn't exceed 10^5. All numbers in the input are positive integers not greater than 10^5.

Output

Print one number for each test case—the length of the longest word that can be cut out at least k times, or 0 if there is no such word.

Examples

standard input	standard output

`4 aaaa`
`1 aaaa`
	`1`
	`4`

`2 cabcxab`
`100 abcaba`
	`3`
	`0`

Problem E. Graph Game

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 second
Memory limit:	128 Mebibytes

Alice and Bob play a game with the following rules:

- Before the game, they draw N vertices without edges.
- On her turn, the player can draw a bidirectional edge between two different vertices that has not yet been drawn. Only one edge is allowed between any pair of vertices.
- The player after whose turn the graph becomes connected loses the game.

Please find out which player wins if both play optimally and Bob moves first.

Input

The first line of input contains the number of test cases T ($1 \le T \le 100$). Next T lines contain one number $N \ge 2$ each which is the amount of vertices in the graph. The total amount of vertices in all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print the name of the winner (“`Alice`” or “`Bob`”) on a separate line.

Examples

standard input	standard output
2 2 3	Alice Bob
1 10	Bob

Problem F. Politicans

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	128 Mebibytes

Members of the Parliament want to make a party before the holidays. It is known that N Tories ans M Whigs will be at the party. They will sit at a circular table with $N + M$ numerated places. Of course, they want you to make a sitting plan. And you want to avoid any political conflicts. It means that there shouldn't be too much Tories or Whigs in one place. More precisely, there should be no more than A consecutive Tories and no more than B consecutive Whigs at the table.

Your task is to find the number of sitting plans without conflicts. You can assume all Tories are indistinguishable (and the same for Whigs).

Input

The only line of input contains four integers N, M, A and B ($1 \leq N, M, A, B \leq 1000$).

Output

Print one integer which is the number of sitting plans without conflicts modulo 10^9.

Examples

standard input	standard output
`5 5 1 1`	`2`
`5 6 1 1`	`0`
`2 3 1 2`	`5`

Problem G. Polyhedron

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	128 Mebibytes

Vasya has strongly decided to go for creating 3D computer models. At first he would like to develop his 3D imagination and to learn how wire frames are constructed. A convex polyhedron was chosen as Vasya's very first model. Vasya asked his friend Petya to choose several points in space and now he wants to create a model that is the convex hull of all the selected points. We will remind you of some definitions:

- A convex set is a set of points in three-dimensional space that together with any two of its points contains the segment connecting these points.
- Convex hull of a set of points is the intersection of all convex sets containing these points.

Your task is to calculate how many straight pieces of wire are needed to create the polyhedron model. The pieces cannot be cut or bent.

Input

The first line of the input file contains an integer number N ($4 \le N \le 300$). This is the number of points that Petya has chosen. Each of the next N lines contains three space-separated integer numbers X, Y, Z which are the coordinates of Petya's points. These numbers do not exceed 1000 by absolute value.

It is guaranteed that the convex hull of all the points specified in the input is a non-degenerate polyhedron (that is, a polyhedron having non-zero volume). Points in the input may coincide.

Output

Output one integer—the amount of wire pieces that Vasya needs.

Examples

standard input	standard output

8
0 0 0
0 0 1
0 1 0
0 1 1
1 0 0
1 0 1
1 1 0
1 1 1
	12

4
0 0 0
0 0 1
0 1 0
1 0 0
	6

Problem H. Traffic Jams

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 second
Memory limit:	128 Mebibytes

The largest GPS navigator manufacturer in a far away kingdom is in an urgent need of a program that would determine optimal paths for automobile transport in a certain city. After a week of painful interface design, reverse engineering of the statistical data format and an exciting study of some "stylish" code (collaborative work is fun, isn't it?) you managed to put the problem as simple as the following.

There are several junctions connected with one-directional roads. Every road has a travel time assigned to it (that is an average time in minutes that you need to pass this road; in your model, everything related to time is rounded to minutes). This value may change during the day due to many reasons (different traffic at different times being the most important). Your analytics specialists suggested the following model. The moment you enter the road from its starting junction determines the time you will spend to pass this road. The main point of the model is that we may assume that the travel time doesn't change over 5-minute intervals (`00:00-00:04`, `00:05-00:09`, etc). The first 5-minute interval starts at midnight, thus we consider that travel times at moments `00:00`, `00:01`, `00:02`, `00:03` and `00:04` are the same and at `00:05` they may change. To avoid large amounts of data to be taken into account you first assumed that travel time change rate (that is, the difference between travel time for consecutive 5-minute intervals) is constant. Luckily, some of your colleagues explained to you that travel time change rate isn't constant at all, and to make up for it you agreed to introduce some key points when travel time change rate changes.

Now you need to find shortest path lengths from two fixed points (client's home and office) to any junction in the city. Junction travel times may be considered negligible.

Input

The first line of input contains two integers: the number of junctions n and the number of roads m ($0 < n \le 10^5$, $0 \le m \le 10^5$). After that, m lines with descriptions of the roads follow, one road per line. The road description starts with four integers a, b, t and q: a and b are indices of junctions connected by the road (drive direction is from a to b), t is the travel time at midnight and q is the number of key points ($1 \le a, b \le n$, $0 \le q \le 10$). The end of the line contains q key points descriptions in ascending order—time of the point in `HH:MM` format (rounded to 5 minutes) and travel time change rate (which comes in effect only 5 minutes later because travel time for a 5-minute interval starting from a key point is calculated using the previous change rate value). After a key point, the travel time change rate becomes exactly the rate given in this key point. Initial change rate (at `00:00`) is 0. Positive change rate means that the travel time increases.

The next line of input contains two integers u and v—the indices of "home" and "office" junctions respectively ($1 \le u, v \le n$).

The next line contains the number of queries T ($T \le 10^4$). Each of the following T lines contains two integers x and y, where x is 0 if the starting point is "home" and 1 if it is "office", and y is the target junction ($1 \le y \le n$). Departure time is always `08:00` for "home" and `18:00` for "work".

You may assume that the key points data are consistent, i. e. the travel time for every road can be uniquely determined at any moment of time. The data obey the "waiting" rule: if $t_1 < t_2$ are two moments of time on an absolute scale and $T(t_1)$, $T(t_2)$ are travel times for a certain road if you depart at t_1 and t_2, then $t_1 + T(t_1) \le t_2 + T(t_2)$. That means you can't complete your journey earlier if you depart later. Note that some journeys may take longer than one day; for any road, the travel time for moment $t +$ `24:00` is the same as the travel time for moment t. The travel time for a road at any moment of time is nonnegative and does not exceed 1000.

At the end of each day, travel time change rate at the moment immediately before midnight will be 0. Still, there can be a key point at `00:00` with non-zero change rate.

Output

For each query, print the time of day for the first moment when you may complete your journey in `HH:MM` format. If you cannot reach your target, print "`Impossible`" (without quotes) instead.

Example

standard input	standard output
4 3 1 2 400 0 2 3 3 0 4 3 10 3 07:00 2 09:00 -1 13:00 0 4 1 2 1 3 0 3	00:43 08:34

Problem I. Robots

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 second
Memory limit:	128 Mebibytes

One famous company has a rectangular warehouse of size $m \times n$ meters. It is split into rectangular cells of size 1×1 meter. There are several automatic robots, each occupying one cell, that can move inside the warehouse. A robot can move to an adjacent cell (the cells are adjacent if they have a common edge). Some cells have obstacles inside them, and robots can't move into these cells.

The wise CEO wants to use several robots, each robot should have its own route. Routes shouldn't intersect, and every route should be a cycle without self-intersections having length greater than two. Every cell without an obstacle must be a part of some route. Write a program that will find out if such a set of routes exists.

Input

The first line of input contains two integers m and n ($1 \leq m, n \leq 50$).

Next m lines contain n integers each. The j-th integer of i-th of these lines will be 1 if the cell contains obstacle and 0 if it is empty.

Output

Print "`Yes`" if the warehouse can be entirely covered with such routes and "`No`" otherwise.

Examples

standard input	standard output

3 3
1 0 0
0 0 0
0 0 0
	Yes

3 3
1 0 0
0 0 0
0 0 1
	No

Problem J. Trees

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	128 Mebibytes

A tree (connected undirected graph without cycles) is given.

Moreover, it's a planar tree, i. e. a tree that is drawn on a plane. Vertices are points, edges are segments that can intersect only at their endpoints.

Two planar trees are called *isomorphic* if one can be transformed into another by a continuous deformation. During the transformation, the planarity properties of the tree must persist.

Your task is to find the maximal number of vertices in a common planar subtree of two given planar trees.

Input

The input contains description of two planar trees. Each tree is described as follows. The first line of the description contains the amount of vertices N ($1 \leq N \leq 200$). Next N lines contain descriptions of edges; i-th of these lines contains the numbers of vertices connected with i-th vertex. Vertices are numbered starting from 1. Each list of vertices is terminated by a single zero. The edges from each vertex are drawn on the plane in the same order as given in the input file.

The degree of each vertex in both of the given trees does not exceed 50.

Output

Print a line with one number—the amount of vertices in a maximal common planar subtree.

Examples

standard input	standard output

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

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

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

Problem K. Triangle and Circle

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	128 Mebibytes

"Ho-ho-ho! I've heard that you like to solve geometry problems. Here is a nice one. Merry Christmas!" said Santa Claus on Christmas Eve.

The problem was to calculate the area of the intersection of a triangle and a circle.

Input

The first line of input contains three integers x_0, y_0 and r_0 which are the coordinates of the center of the circle and its radius. The second line contains six integers x_1, y_1, x_2, y_2, x_3 and y_3 which are the coordinates of the vertices of the triangle. All these integers are positive and do not exceed 100.

Output

On the first line of output, print one real number which is the area of the intersection with absolute or relative error not greater than 10^{-6}.

Examples

standard input	standard output

`1 10 10`
`1 1 1 2 2 1`
	`0.5000000000`

`10 10 10`
`1 1 1 2 2 1`
	`0.0000000000`