Problem A. Analogous Sets

Input file:	`analogous.in`
Output file:	`analogous.out`
Time limit:	2 seconds
Memory limit:	512 mebibytes

For a set A of positive integers let us call $A + A$ a multiset $\{x + y \mid x, y \in A, x \neq y\}$.

Consider two sets A and B of the same size n containing positive integers. A and B are called *analogous* if $A + A$ and $B + B$ are the same multisets. For example, $\{1, 4\}$ and $\{2, 3\}$ are analogous, because $A + A = B + B = \{5\}$, but $\{1, 2, 5, 6\}$ and $\{1, 3, 4, 6\}$ are not, because $A + A = \{3, 6, 7, 7, 8, 11\}$ and $B + B = \{4, 5, 7, 7, 9, 10\}$.

Given n you have to find two analogous sets of size n or detect that there are none.

Input

The input file contains multiple test cases, one on a line.

Each test case is an integer n on a line by itself ($2 \leq n \leq 1000$).

The last test case is followed by a zero that should not be processed.

Output

For each test case print "`Yes`" if there exist two different analogous sets of size n, or "`No`" if there are none. If there exist such sets, the following two lines must contain n positive integers each and describe the found sets.

If there are several possible pairs of analogous sets for some n, you can output any one.

Examples

analogous.in	analogous.out
2 3 0	Yes 1 4 2 3 No

Problem B. Bayes' Law

Input file:	bayes.in
Output file:	bayes.out
Time limit:	2 seconds
Memory limit:	512 mebibytes

Bayes' Law is one of the central theorems of elementary probability theory. It allows to update probability estimations based on experiments. Consider random events A and B. Let A be a positive outcome of an experiment and B be a hypothesis. The probability $P(A|B)$ is the probability that A is observed if B is true. We have $P(A|B) = P(A \cap B)/P(B)$. If we observe A indeed, we can estimate the probability of B as $P(B|A) = P(A|B) \cdot P(B)/P(A)$. In this problem you are given an experiment and the required confidence α, and you must find the best hypothesis B, such that $P(B|A) \geq \alpha$. Let us get into more details now.

Consider a real-valued random variable ξ distributed uniformly between 0 and x. The experiment consists of evaluation of a given function f on the value of ξ. The result of the experiment is considered positive if $a \leq f(\xi) \leq b$. Given f as a piecewise-linear continuous function, a, b, and α, you must find such segment $[L, R]$ that $0 \leq L < R \leq x$, the probability $P(L \leq \xi \leq R \,|\, a \leq f(\xi) \leq b)$ is at least α, and segment length $R - L$ is minimal possible.

Input

The input file contains multiple test cases.

Each test case starts with an integer n — the number of segments in the description of f ($1 \leq n \leq 100\,000$). The following line contains two real numbers: a and b ($0 \leq a < b \leq 10^3$). The following line contains real number α ($0 < \alpha < 1$). Each number is given with at most 3 digits after decimal point, $b - a \geq 10^{-3}$.

After that $n+1$ lines follow, they describe the break points of f's graph. These lines contain two integers each: coordinates of points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, $0 = x_0 < x_1 < \ldots < x_n = x \leq 10^6$, $0 \leq y_i \leq 10^3$. The graph of the function f consists of segments $(x_0, y_0) - (x_1, y_1)$, $(x_1, y_1) - (x_2, y_2)$, etc. It is guaranteed that $P(a \leq f(\xi) \leq b)$ is at least 10^{-3}.

The last test case is followed by a zero that should not be processed. The sum of n in all test cases in one input file is at most 100 000.

Output

For each test case two real numbers: L and R. Your answer may have absolute or relative error of at most 10^{-6} in both $P(L \leq \xi \leq R \,|\, a \leq f(\xi) \leq b) \geq \alpha$ condition and $R - L$ minimization. Tests are designed in such way that there are no segments $[L', R']$ such that $R' - L' < (R - L)(1 - 10^{-6})$ but $P(L' \leq \xi \leq R' \,|\, a \leq f(\xi) \leq b) \geq \alpha - 10^{-6}$.

If there are several possible solutions, output any one.

Examples

bayes.in	bayes.out
6 3.0 5.0 0.9 0 2 2 5 5 0 7 2 8 1 13 6 15 0 0	1.0 13.813333333333333

Problem C. Catalian Sequences

Input file:	catalian.in
Output file:	catalian.out
Time limit:	2 seconds
Memory limit:	512 mebibytes

Consider a sequence $\langle a_1, a_2, \ldots, a_n \rangle$ of non-negative integers. Accent in a sequence is a pair of adjacent elements such that the element with greater index has greater value. For example, there are two accents in sequence $\langle 0, 2, 3, 1, 0 \rangle$: $a_1 = 0$ to $a_2 = 2$ and $a_2 = 2$ to $a_3 = 3$. Let us denote the number of accents among the first k elements of the sequence as A_k. In the given example $A_1 = 0$ $A_2 = 1$, $A_3 = 2$, $A_4 = 2$ and $A_5 = 2$.

A sequence is called *accented* if $a_1 = 0$ and for each i the inequality $a_i \le A_{i-1} + 1$ is satisfied. For example, the sequence $\langle 0, 2, 3, 1, 0 \rangle$ is not accented because $a_2 = 2$ and $A_1 = 0$. The sequence $\langle 0, 1, 0, 2, 3 \rangle$ is in turn accented because $A_1 = 0$, $A_2 = 1$, $A_3 = 1$, $A_4 = 2$.

A sequence $\langle a_1, a_2, \ldots, a_n \rangle$ of non-negative integers is called *catalian* if the following conditions are satisfied:

1. $\langle a_1, a_2, \ldots, a_n \rangle$ is accented;
2. there are no i, j and k such that $1 \le i < j < k \le n$ and $a_k < a_i < a_j$.

For example, the sequence $\langle 0, 1, 0, 2, 3 \rangle$ is catalian, as well as the sequence $\langle 0, 1, 0, 2, 1 \rangle$ is, but the sequence $\langle 0, 1, 0, 2, 0 \rangle$ is not catalian because for $i = 2, j = 4, k = 5$ we have $a_k = 0 < a_i = 1 < a_j = 2$.

Given n find the number of catalian sequences of length n. For example, if $n = 3$ there are 5 such sequences: $\langle 0, 0, 0 \rangle$, $\langle 0, 0, 1 \rangle$, $\langle 0, 1, 0 \rangle$, $\langle 0, 1, 1 \rangle$, $\langle 0, 1, 2 \rangle$.

Input

The input file contains multiple test cases, one on a line.

Each test case is an integer n on a line by itself ($1 \le n \le 32$).

The last test case is followed by a zero that should not be processed.

Output

For each test case output one line containing the case number and the number of catalian sequences of length n. Adhere to the format of sample output.

Examples

catalian.in	catalian.out
1 2 3 4 5 0	Case #1: 1 Case #2: 2 Case #3: 5 Case #4: 14 Case #5: 42

Problem D. Drunkard's Walk

Input file:	drunkard.in
Output file:	drunkard.out
Time limit:	2 seconds
Memory limit:	512 mebibytes

Once Denis heard the following legend.

A drunkard is walking randomly in a directed graph G with n vertices numbered from 1 to n. Each vertex except vertices $n-1$ and n has exactly two outgoing edges. The drunkard starts at vertex 1. Each second he chooses randomly uniformly one of the two outgoing edges and walks along it. He finishes his walk either in his home at vertex $n-1$, or in the bar at vertex n. The probability that the drunkard ends his walk at home is exactly p/q.

Now Denis wonders, what the graph G could be.

Help him, find such graph G that the story above is true.

Input

The input file contains multiple test cases, one on a line.

Each line of the input file contains two integers: p and q ($1 \le p < q \le 100$).

The last test case is followed by two zeroes that should not be processed.

There are at most 200 test cases in one input file.

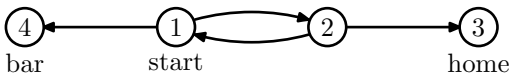
Output

For each test case output the description of the graph G. The first line must contain n — the number of vertices. The number of vertices must be at most 1000. Each of the following $n-2$ lines must describe edges going out from the corresponding vertex. The i-th of these lines must contain two integers: u_i and v_i — the numbers of vertices where the edges from the i-th vertex go to. The graph can have multiple edges from one vertex to another if needed. The edge can go from a vertex to itself if needed.

Examples

drunkard.in	drunkard.out
1 3 0 0	4 2 4 3 1

The graph in the example is shown on the picture below.



Problem E. Elegant Scheduling

Input file:	elegant.in
Output file:	elegant.out
Time limit:	5 seconds
Memory limit:	512 mebibytes

Eddy is developing schedule for his new project. There are $n = 2^k$ jobs to be completed, numbered from 0 to $n-1$. Completing a job takes one day, Eddy's worker Eve will use the following 2^k days to complete the jobs. They have agreed on the following payment scheme: if the job j is completed next day after job i, Eddy must pay $d_{i,j}$ dollars to Eve.

Eddy would like to order jobs in a way to pay Eve as few as possible. But he knows that the problem of finding the best possible ordering is NP-complete, so he decides to choose the best *elegant* ordering. Elegant ordering is composed using the following algorithm. Initially all jobs are ordered by their numbers: $0, 1, 2, \ldots, n-1$. In one step the algorithm divides the sequence of jobs $a_0, a_1, \ldots, a_{2^i-1}$ to its first half $a_0, a_1, \ldots a_{2^{i-1}-1}$ and its second half $a_{2^{i-1}}, \ldots a_{2^i-1}$. Eddy may choose to complete halves in this order, or swap them. After that, if $i > 1$, the same algorithm is applied to each of the halves.

For example, ordering $1, 0, 2, 3, 7, 6, 5, 4$ is elegant, because it can be obtained from the initial ordering $0, 1, 2, 3, 4, 5, 6, 7$ by the following sequence of steps (the considered part of the sequence is in brackets):

- $[0, 1, 2, 3, 4, 5, 6, 7]$: do not swap halves
- $[0, 1, 2, 3], 4, 5, 6, 7$: do not swap halves
- $[0, 1], 2, 3, 4, 5, 6, 7$: swap halves to get $1, 0, 2, 3, 4, 5, 6, 7$
- $1, 0, [2, 3], 4, 5, 6, 7$: do not swap halves
- $1, 0, 2, 3, [4, 5, 6, 7]$: swap halves to get $1, 0, 2, 3, 6, 7, 4, 5$
- $1, 0, 2, 3, [6, 7], 4, 5$: swap halves to get $1, 0, 2, 3, 7, 6, 4, 5$
- $1, 0, 2, 3, 7, 6, [4, 5]$: swap halves to get $1, 0, 2, 3, 7, 6, 5, 4$

However, the ordering $1, 2, 0, 3, 7, 6, 5, 4$ is not elegant since it cannot be obtained from the initial ordering by these rules.

Given n and a way to generate $d_{i,j}$ find the minimal possible sum Eddy can pay to Eve and the order the jobs must be completed.

Input

The input file contains multiple test cases.

The first line of each test case contains an integer n ($2 \le n \le 4096$, n is power of 2). The following line contains a, b, c and m ($0 \le a, b, c \le 10^5$, $2 \le m \le 10^5$). You can calculate values of $d_{i,j}$ for i and j from 0 to $n-1$ using the following formula: $d_{i,j} = (ai + bj + c(i \oplus j)) \bmod m$ where $x \oplus y$ is bitwise exclusive or of x and y (for example, $13 \oplus 7 = 1101_2 \oplus 0111_2 = 1010_2 = 10$).

The last test case is followed by a zero that should not be processed. The sum of n in all test cases doesn't exceed 4096.

Output

For each test case print two lines. The first line must contain the minimal total sum Eddy can pay to Eve. The second line must contain the order the jobs should be completed. If there are several solutions, output any one.

Examples

elegant.in	elegant.out
8 6 5 7 10 0	27 5 4 7 6 1 0 2 3

Problem F. Flights

Input file:	`flights.in`
Output file:	`flights.out`
Time limit:	2 seconds
Memory limit:	512 mebibytes

Farcian Federation is a big country and its n cities are located at great distance from each other. Therefore the main way to move between cities there is making a flight by a plane. There are m bidirectional flights connecting different cities. For each city there is a flight connecting it with Cowmos, the capital of Farcian Federation, and there can be some other flights.

The new minister of transportation of Farcian Federation is planning to make several reforms to the Farcian aviation to counter terrorism. The first reform he is planning is changing flight numbers.

After the reform each flight must have a number from 1 to m. Different flights must have different numbers. For any two different cities u and v the sum of numbers of flights connecting u to other cities must be different from the sum of numbers of flights connecting v to other cities.

Help the minister to choose new numbers for the flights so that the condition above was satisfied.

Input

The input file contains multiple test cases.

The first line of each test case contains n and m — the number of cities and the number of flights, respectively ($3 \le n \le 1000$, $n - 1 \le m \le 100\,000$). Let the cities of the country be numbered from 1 to n and let the capital have number 1. The following m lines describe flights. Each flight is described by two integers u_i and v_i — the numbers of the cities it connects. No two cities are connected by more than one flight. No city is connected by a flight to itself. Each city is connected by a flight to city 1.

The last test case is followed by two zeroes that should not be processed.

The sum of n in all test cases in one file doesn't exceed 100 000. The sum of m in all test cases in one file doesn't exceed 100 000.

Output

Print answers to all test cases.

If it is possible to give numbers to flights so that the described condition is satisfied, output "`Yes`" as the first line of output. In the other case output "`No`".

If the numbering is possible, the following line must contain m integers: for each flight in order they are given in the input file print its new number. If there are several solutions, output any one.

Examples

flights.in	flights.out
5 8 1 5 1 3 1 4 1 2 2 3 3 4 4 5 5 2 0 0	Yes 1 3 2 4 5 6 7 8

In the example the sums of flight numbers in the cities are 10, 17, 14, 15, and 16, respectively.

Problem G. Genome of English Literature

Input file:	`genome.in`
Output file:	`genome.out`
Time limit:	2 seconds
Memory limit:	512 mebibytes

Genome assembly is an important problem in bioinformatics. Genome is a very long string, so reading it completely from DNA is a difficult task. To extract genome information so called *sequencing machines* are used. They take DNA and split it to small pieces. After that these pieces are scanned to get so called *pair reads* — prefix and suffix of each piece of some length k are recorded. Some characters in reads can be scanned incorrectly, they correspond to *read errors*. Usually several copies of the same genome are sequenced, thus providing *multiple cover* of the genome. Genome assembly problem is then to restore genome from these reads.

In this problem we would attempt to develop genome assembly algorithms and apply them to classic English literature. We will ignore pairness of reads and will not introduce any errors, so your problem will be simpler than that of bioinformatic scientists.

Judges took 12 classic English and American literature texts as test data:

- William Shakespear — "Romeo and Juliet" (sample test)
- Daniel Defoe — "Robinson Crusoe"
- Jonathan Swift — "Gulliver's Travels"
- Jack London — "White Fang"
- The Works Of Edgar Allen Poe
- Matthew Lewis — "The Monk"
- Arthur Conan Doyle — "The Hound of the Baskervilles"
- Charles Dickens — "Great Expectations"
- H.G. Wells — "The War of the Worlds"
- Herman Melville — "Moby Dick or The Whale"
- Mark Twain — "The Adventures of Tom Sawyer"
- Horace Walpole — "The Castle of Otranto"

Each of these texts was downloaded from Project Guttenberg library as plain text. First it was converted to a sequence of characters with ASCII codes from 32 to 126. All characters with ASCII code less then 32 (space) were replaced with space and all characters with ASCII code greater than 126 were removed. All sequences of two or more consecutive spaces were replaced with one space. All characters except the first 50 000 were removed. Let us call the resulting string t.

After that 20 000 times random integer i from 1 to 49 951 was uniformly generated and 50 characters at positions $t[i \ldots i + 49]$ were printed to the input file as one line. Therefore input file contains 20 000 lines of 50 characters each, they represent some random subwords of length 50 of t.

Your task is to cover a great part of t with *scaffolds*. For the purpose of this problem scaffold is a string of length at least 500 that is a substring of t. You have to print one or more words to the output file with the total length of at most 50 000. Words that are not substrings of t will be ignored. For words that are long enough substrings of t all occurrences will be found. Positions in t that are covered by these occurrences will be marked. Your output for a test will be accepted if at least half of positions of t are marked.

Input

The input file contains 20 000 strings of length 50. Each string is a random substring of a text t of length 50 000 obtained as described in the problem statement.

Output

Output one or more words that you think are substrings of t of length at least 500. Words must have total length of at most 50 000.

Your output will be accepted if at least half of positions in t are covered by occurrences of words you print.

Examples

You can download sample input/output from `http://forest.acm.petrsu.ru`

Problem H. Hide-and-Seek (Division 1 Only!)

Input file:	`hide.in`
Output file:	`hide.out`
Time limit:	2 seconds
Memory limit:	512 mebibytes

Little Henry likes to play hide-and-seek with his friends. But Henry doesn't like conventional rules of the game, so he has invented his own rules.

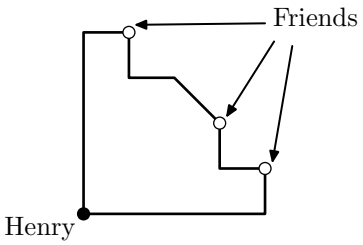
The boys play in Henry's room which has a form of a simple polygon with n corners numbered from 1 to n in counterclockwise order. Let us say that the point A of the room is visible from the point B if the segment AB is completely inside the room (it is allowed to touch room walls or corners).

The room has such form that the following conditions are satisfied:

- the first corner is *convex*: that is, the angle between the two walls adjacent to the first corner is less than 180° when measured inside the room;
- all other corners of the room are visible from the first corner.

Henry stands in the first corner of the room and his friends then choose other corners of the room in such way that none of them can see each other and hide there. After that friends try to guess who have chosen which corner, and seeing them all Henry has fun.

The picture below shows the room from the first example with Henry and three friends playing.



Henry would like to play this funny game with as many friends as possible. Help him to find out how many friends he can invite to play so that they could choose corners to hide in.

Input

The input file contains multiple test cases.

The first line of each test case contains n — the number of corners of Henry's room ($3 \le n \le 500$). The following n lines contain coordinates of room corners, described in counterclockwise order. Each line contains two integers x_i and y_i ($-10^5 \le x_i, y_i \le 10^5$).

The last test case is followed by a zero that should not be processed.

The sum of n in all test cases in one file doesn't exceed 500.

Output

Print answers to all test cases.

For each test case first print k — the maximal number of friends Henry can invite. After that print k integers: numbers of corners the friends can choose to hide. The corners must not be visible from each other. Corners are numbered from 1 to n in order they are given in the input file. Corner 1 must not be chosen because Henry is there.

If there are several solutions, print any one.

Examples

hide.in	hide.out
9 0 0 4 0 4 1 3 1 3 2 2 3 1 3 1 4 0 4 4 0 0 1 0 1 1 0 1 0	3 3 5 8 1 2

Problem I. Informatics Final Projects (Division 1 Only!)

Input file:	`informatics.in`
Output file:	`informatics.out`
Time limit:	2 seconds
Memory limit:	512 mebibytes

Ivan is the head of Informatics Department in Innopolice Institute for Investigations and Innovation. Now it's time for the students of the department to select their final project, and Ivan is going to distribute projects among students.

There are n students in the department, m available projects and t teachers going to supervise the projects. Each student has chosen one or several projects that he would agree to work on and ordered them in a list by his preference. The first project in the list is the project students would to work on most of all, the second project is a bit worse, but still acceptable, and so on. Projects not on the student's list should not be assigned to him. Each project is on the list of at least one student.

Each teacher has chosen one or more projects he is ready to supervise. Each project was chosen by exactly one teacher. There are also capacity limitations: at most p_i students are allowed to work on the i-th project and the j-th teacher is able to supervise at most t_j students. After choosing projects each teacher ordered all students that agree to take at least one of the projects he would supervise in a list by his preference. The most preferred student is the first one on the list, and so on.

Now Ivan has a hard problem of choosing projects assignment for students. He calls an assignment of students to projects *good* if the following conditions are satisfied:

- Each student is assigned to a project on his preference list, or assigned to no project.
- For each i the i-th project is assigned to at most p_i students.
- For each j the j-th teacher has at most t_j students assigned to projects he supervises.
- There is no pair student s – project x, such that s is not assigned to x and if we assign s to x, it would make the assignment *better*.

We say that assigning student s to project x makes assignment A *better* if the following conditions are satisfied:

- Student s is not assigned to any project in A, or is assigned to a project that goes after x in his preference list.
- The project x has less then p_x students assigned to it, or the teacher that supervises x has s on his preference list before at least one of the students that are assigned to x in A.
- The teacher u who supervises the project x has less then t_u students assigned to all his projects, or he has s on his preference list before at least one of the students that are assigned to one of his projects in A.

To start with Ivan would like to find any good assignment of students to projects. Help him to find it.

Input

The input file contains multiple test cases.

Each test case starts with three integers n, m and t ($1 \le n \le 100$, $1 \le t \le m \le 100$).

The following n lines describe students. The i-th of these lines starts with k_i — the number of projects in the i-th student's preference list, followed by k_i distinct integers from 1 to m — the list itself, from the most preferrable project to the least preferrable but still acceptable.

The line with project capacities $p_1, p_2, \ldots, p_m$ follows ($1 \le p_i \le n$).

After that t teacher descriptions follow. Each teacher description consists of three lines. The first of these lines contains one integer t_j — the maximal number of students the teacher accepts to supervise ($1 \le t_j \le n$). The second line contains l_j — the number of students on the teacher's preference list followed by the list itself — l_j distinct integers from 1 to n, from the most preferrable student to the least preferrable. The third line starts with z_j — the number of projects the teacher is going to supervise followed by the list of these projects ($1 \le z_j \le m$).

Each project is supervised by exactly one teacher. Each teacher lists exactly those students in his preference list that accept to take at least one of his projects.

The last test case is followed by three zeroes that should not be processed.

The total number of students in all test cases in one input file doesn't exceed 1000. The total number of projects in all test cases in one input file doesn't exceed 1000. The total number of teachers in all test cases in one input file doesn't exceed 1000.

Output

Print answers to all test cases.

For each test case print one line with n integers: any good assignment of projects to students. For each student print the number of the project it must be assigned to, or 0 if the student should not be assigned to any project.

If there are several solutions, print any one. It can be proven that at least one good assignment always exists.

Examples

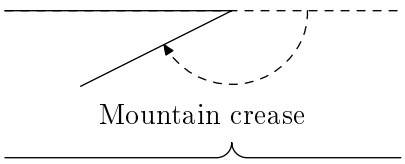
informatics.in	informatics.out
7 8 3 2 1 7 6 1 2 3 4 5 6 3 2 1 4 1 2 4 1 2 3 4 5 2 3 4 5 6 3 5 3 8 2 1 1 1 1 1 1 1 3 7 7 4 1 3 2 5 6 3 1 2 3 2 5 3 2 6 7 5 3 4 5 6 2 2 1 7 2 7 8 0 0 0	1 5 4 2 0 0 3

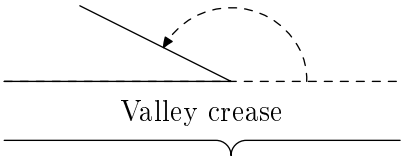
Problem J. Japanese Origami (Division 1 Only!)

Input file:	`japanese.in`
Output file:	`japanese.out`
Time limit:	2 seconds
Memory limit:	512 mebibytes

Jeremy visits Japanese Origami club after school. His study of origami art has just started, so the first task given by his teacher is to fold the strip with creases so that all folds were along the creases and matched their types.

Consider a paper strip. There are two ways to fold it which creates two possible types of *crease*. The picture on the left shows *mountain* crease which is created when the segment of the strip on the right is placed *under* the segment on the left. The picture on the right shows *valley* crease which is created when the segment of the strip on the right is placed *above* the segment on the left.





The paper strip was folded several times to create some creases. Each fold could use some of the layers, not necessarily all or only one. Segments of the strip could be bent or curved during folds, but the paper was not torn and after all folds it was compeletely flat, folded at all creases and only there.

You are given the description of the strip after complete folding and then unfolding: the sequence of segment lengths between creases and crease types. You must detect whether it was possible to fold the strip in such a way and if it was possible how the folded strip could possibly look. You must assume that the strip is infinitely thin and creases are infinitely small.

Input

The input file contains multiple test cases.

The first line of each test case contains n — the number of creases on the paper strip ($1 \le n \le 500$). The following line contains $2n + 1$ tokens: $l_0, c_1, l_1, c_2, l_3, \ldots, c_n, l_n$. Here l_i are integers, $1 \le l_i \le 10^5$, l_0 is distance from the strip endpoint to the first crease, l_1 is the distance between the first and the second crease, etc, c_i describe corresponding crease types and are either '`M`' or '`V`' for mountain and valley-type creases, respectively. Creases are described from left to right.

The sum of n in all test cases in one file doesn't exceed 5 000.

Output

Print answers to all test cases.

The first line of the answer must be either "`Yes`" if it is possible to fold the strip so that all creases were correct, or "`No`" if it is impossible.

If it is possible to fold the strip, the following line must contain $n+1$ numbers and describe layer structure of the folded strip. You must assign distinct integers from 1 to $n+1$ to all segments of the strip from the input so that the following condition was satisfied. Let us position the folded strip along the line so that the first desribed segment (with length l_0) extends to the right from the endpoint of the strip. For each two segments X and Y if at some point there are both segments X and Y, and X is is above Y, then X must be assigned smaller number than Y.

If there are several solutions, print any one.

Examples

japanese.in
2 3 M 4 M 3 2 4 M 3 M 4 2 4 M 3 V 4 11 7 M 4 M 3 V 4 V 4 M 4 V 5 V 4 M 7 V 7 V 7 M 8 0

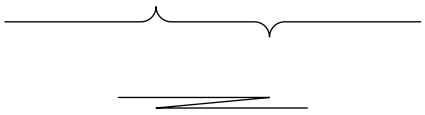
japanese.out
Yes 2 3 1 No Yes 1 2 3 Yes 3 6 5 4 8 9 12 11 10 1 2 7

Notes

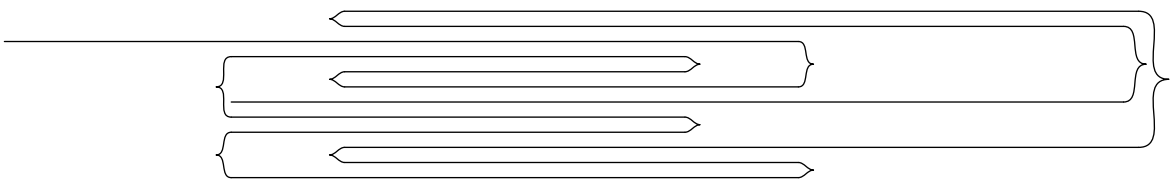
The picture below shows how the strip from the first example can be folded. The top picture shows the strip with creases, the bottom picture shows the folded strip right before it is made completely flat.

The strip from the second example cannot be folded flat to create such creases.

The picture below shows how the strip from the third example can be folded. The top picture shows the strip with creases, the bottom picture shows the folded strip right before it is made completely flat.



Finally, the picture below shows schematically one way to fold the strip from the fourth example (there are also many other ways). To make clear view it shows some distance between layers, actually layers are infinitely thin and there is no distance between them. Creases are also shown as having some non-zero size, actually they are infinitly small.



Problem K. Kabbalah for Two (Division 1 Only!)

Input file:	kabbalah.in
Output file:	kabbalah.out
Time limit:	5 seconds
Memory limit:	512 mebibytes

Kai and Kevin are kabbalists. However, they don't like traditional kabbalah with its pentagrams, etc. They have developed their own version of kabbalah that has its rituals centered around circles.

Now they are preparing a place for their kabbalistic rituals. They are going to have rituals at the backyard of Kevin's grandmother's house. The backyard has a form of a convex polygon with n vertices. Friends need to place two circular mats to the yard for their rituals. The mats must fit completely into the yard without overlapping. Of course none of the friends would agree to have a smaller mat, so their radii must be equal to each other.

To have as powerful rituals as possible, Kai and Kevin would like to have as big mats as possible. Help them to find out what maximal radius of the mats could be.

Input

The input file contains multiple test cases.

The first line of each test case contains n — the number of vertices of the backyard polygon ($3 \le n \le 200$).

The following n lines describe vertices of the polygon in counterclockwise order, each vertex is described by its integer coordinates x_i and y_i ($-10^4 \le x_i, y_i \le 10^4$). The given polygon is convex, no three vertices are on the same line.

The sum of n in all test cases in one file doesn't exceed 200.

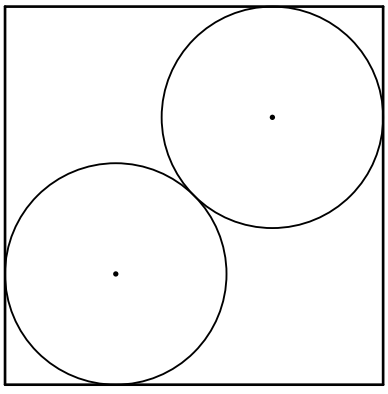
Output

For each test case first output one floating point number — the maximal possible radius of two non-overlapping circular mats of the same size that can be put to the backyard. The following two lines must contain two floating point numbers each — coordinates of centers of mats.

Your answer must have absolute or relative error of at most 10^{-6}.

If there are several optimal solutions, output any one.

Examples

kabbalah.in	kabbalah.out	Notes
4 0 0 10 0 10 10 0 10 0	2.9289321881345248 2.9289321881345248 2.9289321881345248 7.0710678118654752 7.0710678118654752	

Problem L. Legendary Boomerang Competition (Division 2 Only)

Input file:	`legendary.in`
Output file:	`legendary.out`
Time limit:	2 seconds
Memory limit:	512 mebibytes

In the Legendary Boomerang Competition, a competitor can throw a boomerang to any location, after which it will return to him in a straight line and knock out every target in its path. The initial throw is harmless, as the sport boomerang hits the targets only on the return path. Given the locations of each target in the room relative to competitor, determine the minimum number of boomerangs he will have to hit each target. Note that competitor must catch the boomerang once it returns, as letting it continue its flight would be penalized by immediate disqualification.

Input

The first line of the input file will contain a single positive integer, T, the number of testcases in the playing card factory (). A single test case will begin with a single positive integer, n ($1 \le n \le 100$), on a line by itself, representing the number of targets. Following this will be n line s in the format x y ($-1000 \le x, y \le 1000$, x and y will never both be zero), representing the x and y coordinates of a target relative to the competitor.

Output

If competitor will need more than one boomerang shoot to hit all targets, output "`X shoots!`", where x is the smallest number of shoots he can use. If competitor can hit all targets with a single boomerang, instead output "`Great! n targets with one shot!`", where n is the number of targets.

Example

legendary.in	legendary.out
3 4 1 1 2 2 1 0 5 0 3 1 1 -3 -3 8 -4 4 -1 1 -7 7 -49 49 -343 343	2 shoots! 3 shoots! Great! 4 targets with one shot!

Problem M. Monkey and the Broken Typewriter (Division 2 Only!)

Input file:	monkey.in
Output file:	monkey.out
Time limit:	2 seconds
Memory limit:	512 mebibytes

Well trained monkey is trying to print some words from dictionary using broken typewriter. Given a dictionary of words and the text, printed by monkey, output a list of possible words the monkey was possibly trying to type. Remember that broken typewriter tend to drag out each letter of a word, that even well trained keyboard can type letters in the word in any order.

So, two words are equivalent if they have the same number of maximal *runs* of each letter. A run is a sequence of consecutive identical characters in a word. A maximal run is simply a run whose adjacent characters differ from its contents.

For example, word "CAREER" is composed of 5 maximal runs ("C", "A", "R", "EE", "R"). "AAARRRCCCCCEEEERRR" also breaks into 5 maximal runs ("AAA", "RRR", "CCCCC", "EEEE", "RRR"). Since both words have an equal number of runs of each letter (one 'A', one 'C', one 'E', 2 'R and 0 for all other letters), they are a match.

Input

Input file will begin with a single positive integer, t, the number of test cases ($1 \le t \le 15$). A testcase begins with a line containing a single positive integer, d ($d \le 1000$), the size of the dictionary, which is in turn followed by d distinct words, each on its own line. A word consists of anywhere from 1 to 30 uppercase alphabetic characters. Following the dictionary is a line containing a single positive integer, q ($q \le 100$), the number of texts, typed by monkey.

Following this are q lines, each containing a single text. A text consists of anywhere from 1 to 100 uppercase alphabetic characters.

Output

For each text in this scenario, if it could not be unscrambled using the dictionary print "No matches found." Otherwise, print "Did you mean:" on its own line, followed by a list of possible matchings sorted in alphabetical order, each followed by a single '?'. Print each matching on its own line and in upper case.

Examples

monkey.in	monkey.out
1 8 CAREER CARER DEER DREER RACE RACECAR RACER RED 3 AAAARRRCCCCEEEERRRR PSYCHOLINGUISTICS DER	Did you mean: CAREER? CARER? RACER? No matches found. Did you mean: DEER? RED?

Problem N. Need For Souvenirs (Division 2 Only!)

Input file:	nfs.in
Output file:	nfs.out
Time limit:	2 seconds
Memory limit:	512 mebibytes

On the final of some popular programming contest a line of N contestants forms to receive souvenirs. Sponsor has Y different types of souvenirs and exactly B souvenirs of each type. The first contestant in line receives a souvenir of the first type, the second person receives a souvenir of the second type, and so on.

When the sponsors give out a souvenir of the last type, they start again with the first type. If they reach the end of the line, they begin again with the first person in line, being careful to keep track of which type they had just given out. This process continues until there are no more souvenirs.

Knowing the single type of souvenir S that you like most, you want to figure out where in line you should stand in order to obtain the second-most souvenirs of that type. After all, it would look suspicious if you were to always receive the most. Although if there are multiple positions in line which receive the most of that type, then you're okay with receiving the most, and of those positions you'd like the one closest to the end of the line since it's easier to get. Same for the case of multiple positions receiving second-most.

Input

The first line is the number T of testcases ($1 \le T \le 15$). Each testcase a single line containing the integers N, Y , B, and S. Each integer is at least 1 and at most 100. In addition, $N \ge 2$ and $S \le Y$.

Output

For each testcase output the position in line at which you should stand.

Example

nfs.in	nfs.out
2 4 6 3 5 4 3 6 1	3 4

Problem O. Operation X (Division 2 Only!)

Input file:	operationx.in
Output file:	operationx.out
Time limit:	2 seconds
Memory limit:	512 mebibytes

The security agency have a keen interest in tracking meta-data of phone calls in order to build a network of who seems to be friends with whom. In fact, they claim that the content of phone calls has not been spied upon, and only the meta-data (who called whom when and for how long) is being logged. The idea is that calling patterns will reveal social connections.

In this question, you will be given a network of who is friends (or associated) with whom, and the identity of one suspicious person — mr. X. Your task is to output all friends of that person in the network, presumably for further investigation.

Input

The first line is the number T of input data sets, followed by the T data sets ($1 \le T \le 20$), each of the following form: The first line contains two integers n, m, separated by a space. $1 \le n \le 100$ is the number of people in your network. $0 \le m \le 10^4$ is the number of friendships among pairs of people. Next come m lines, each containing two distinct integers between 1 and n, describing a connection between those two people.

Finally, on one more line, is a number s, $1 \le s \le n$, the identity of mr. X.

Output

For each data set, output on one line all the direct friends of person s in the network, sorted in increasing order, separated by a space. (No person is friends with himself/herself, and we are not interested in friends-of-friends etc). If list is empty, print empty string instead.

Example

operationx.in	operationx.out
2 3 2 2 3 3 2 1 5 5 1 2 4 1 2 4 1 4 3 5 1	2 4