Problem A. Boxes

Input file:	`boxes.in`
Output file:	`boxes.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

There are N boxes. They are numbered with positive integers from 1 to N without repetitions. Additionally, an integer between 1 and N is written on the bottom of each box. Note that there can be several boxes with the same number written on them.

Let us define a *move* as the following sequence of actions:

1. Choose any box and mark it as current.
2. Remember the number written on the bottom and remove the box.
3. If there exists a box with the number just have been remembered, mark that box as current and go to step 2. Otherwise, the move is over.

Given the numbers written on the bottoms of the boxes you should determine what minimum and maximum number of moves one could possibly make to remove all the boxes.

Input

The first line of the input file contains an integer N ($1 \le N \le 10^5$). Each of the following N lines contains one number. The i-th of these lines contains the number written on the bottom of the i-th box.

Output

The output file should contain two integers separated by space — the minimum and maximum number of moves.

Example

boxes.in	boxes.out
4 2 3 4 2	1 2

Problem B. Domino sorting

Input file:	`domino.in`
Output file:	`domino.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Denis, a young programmer, received a set of dominoes as a birthday present. As Denis was a programmer he didn't know how to play domino. That's why he invented a new game: he took N dominoes from the set and made the rectangle $2 \times N$ from them so that one domino piece makes one horizontal row. Then he swapped and turned over some of them in order to make numbers in the left column be sorted upside down in nondecreasing order and numbers in the right column be sorted in nonincreasing order. Denis called this game "domino sorting".

However, this game takes a lot of time... Now Denis wants to write a program that will sort any suggested set of dominoes. But as Denis is a young programmer he got stuck with this problem. So he asked you to help him.

Input

First line of input contains an integer N ($1 \leq N \leq 10^5$). The following N lines describe dominoes. The (i)–th of these lines contains two integers, separated by space — a_i and b_i ($0 \leq a_i, b_i \leq 10^6$). They correspond to numbers on the i–th domino.

Output

First line of output should contain `YES` if the given set of dominoes can be sorted as explained. Next N lines should describe the dominoes in the sorted order — two numbers separated by space in each line. Numbers in the first column should be nondecreasing and from the second column — nonincreasing. If there is more than one solution you may output any of them. In case there is no solution you should a output single line `NO`.

Examples

domino.in	domino.out

3
5 2
6 1
3 4
	YES
	1 6
	2 5
	3 4

4
1 5
7 1
3 8
5 6
	YES
	1 7
	1 5
	6 5
	8 3

2
1 2
3 4
	NO

Problem C. Duty

Input file:	`duty.in`
Output file:	`duty.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

A squad consists of N soldiers. Every day exactly three of them must go on duty.

Captain Evidence started to form a duty chart. But suddenly he encountered with the evident fact that any two solders who were on duty together become mentally closer. If two soldiers were on duty together for three times, then the forth co-duty can make them so mentally close that evidently something really bad would happen.

So the Captain wants to make a duty chart for maximum possible number of days so that any two soldiers have at most three co-duties. Unfortunately it's far from evident to the Captain how to do it, so he hopes that you can help him.

Furthermore, the Captain, willing to become a bit mentally closer with you, told you that there is an odd number of soldiers in the squad.

Input

The first line contains an odd integer N ($3 \leq N \leq 99$) — the number of soldiers in the squad.

Output

The first line should contain one integer K — the maximal number of days. The i-th of the following K lines should contain three numbers a_i, b_i and c_i — numbers of soldiers that should go on duty on the i-th day. Soldiers are numbered with integers from 1 to N.

Examples

duty.in	duty.out

`3`	`3`
	`1 2 3`
	`2 3 1`
	`3 1 2`

`5`	`10`
	`1 2 3`
	`1 2 4`
	`1 2 5`
	`1 3 4`
	`1 3 5`
	`1 4 5`
	`2 3 4`
	`2 3 5`
	`2 4 5`
	`3 4 5`

Problem D. Yet Another Answer

Input file:	`fluffy.in`
Output file:	`fluffy.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Eventually, after running Yet Another Stupendous Supercomputer for π million of years, hedgehogs happen to know Yet Another Answer. This time the Answer they got was quite surprising. Even for hedgehogs who didn't actually know what the Question was.

In fact, the Answer consists of N words. Each word is a nonempty sequence of left and right brackets. Whilst most of the hedgehogs were rather frustrated with the Answer, Fluffy was happy, because Fluffy likes brackets so much. But most of all he likes regular bracket sequences.

A *regular bracket sequence* is a sequence of brackets which satisfies the following conditions:

1. The sequence contains an equal number of left and right brackets;
2. The number of right brackets in any prefix of the sequence doesn't exceed the number of left ones.

Fluffy is sure that it's vitally important for the hedgehogs to know what is the longest regular bracket sequence one can make by concatenating the words of the Answer under the following rules:

1. Each word can be used at most once;
2. Words can be used in the arbitrary order.

Because the Supercomputer is so busy designing its yet more powerful successor, Fluffy hopes you can help him to solve the problem.

Input

The first line contains an integer N ($1 \le N \le 1000$) — the number of words in the Answer. Next N lines contain the words, each on a separate line. The total length of all words doesn't exceed 10000.

Output

The first line of the output should contain two space-separated integers L and K, where L is the length of the longest possible regular bracket sequence and K is the number of words it consists of. The second line should contain K space-separated integers — numbers of used words in the order they should be concatenated. Words are numbered from 1 to N in the order they are given in the input file. If there are several possible answers output any of them.

Examples

`fluffy.in`	`fluffy.out`

`4`
`(`
`(((`
`(`
`))`
	`4 3`
	`1 3 4`

`3`
`()`
`(()`
`)`
	`6 3`
	`2 3 1`

Problem E. Hyperrook

Input file:	`hyperrook.in`
Output file:	`hyperrook.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

A point of N-dimensional Euclidean space is called *valid* if all its coordinates are integers between 0 and $M-1$, inclusive. Thus, there are M^N different valid points. A *hyperrook* can make a *move* from point a to point b if a and b are valid points which differ in exactly one coordinate. For example, $(0,2,1) \to (2,2,1) \to (2,2,0) \to (2,1,0)$ represents a sequence of three moves in three-dimensional space.

A *route* of length d from point t_0 to point t_d is a sequence of valid points $t_0, t_1, ..., t_d$ such that for any i from $\{0, 1, \dots (d-1)\}$ a hyperrook can make a move from point t_i to point t_{i+1}.

Given integers N, M, d, Q and valid points $t_1, t_2, ..., t_Q$ you are to find the number of different routes of length d from t_i to t_j for any pair (i,j) where $1 \le i, j \le Q$.

Input

The first line contains five integers N $(1 \le N \le 50)$, M $(2 \le M \le 10^5)$, d $(0 \le d \le 10^9)$, P $(1 \le P \le 10^9)$ and Q $(2 \le Q \le 50)$. Next Q lines describe points $t_1, t_2, ..., t_Q$. The i-th of these lines contains N integers, each between 0 and $M-1$, inclusive — coordinates of t_i.

Output

Output Q lines each containing Q integers. The j-th number in i-th line should be equal to the number of different routes of length d from t_i to t_j modulo P.

Examples

hyperrook.in	hyperrook.out

2 8 4 10000000 4
3 5
0 5
3 7
0 0
	896 720 720 560
	720 896 560 720
	720 560 896 560
	560 720 560 896

3 3 4 10000000 3
0 2 2
1 1 1
1 2 2
	90 36 45
	36 90 54
	45 54 90

Problem F. Inspection of properties

Input file:	`kingdom.in`
Output file:	`kingdom.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

A long time ago there was a Tzar in Three-binary Kingdom... He had a few years left to live and he decided to make one last journey around his kingdom to see what was going on there.

Kingdom had N towns and each pair of them was connected by exactly one bidirectional road. One of the towns was a capital where the Tzar lived.

The Tzar was very old and his memory often played a bad joke with him. That was why the Tzar decided to make a journey. He was going to visit each town exactly twice, just in case. His journey should begin and end at the same town — the capital.

While the Tzar had been choosing the best option for the journey he encountered an interesting question: how many different routes for the journey exists so that each of the earlier mentioned conditions is met modulo P?

Input

First line of the input contains two integers N and P ($3 \le N \le 10^5$, $1 \le P \le 10^9 + 7$).

Output

Output should contain one number — the answer for the Tzar's question.

Examples

`kingdom.in`	`kingdom.out`
`3 123`	`2`
`4 123`	`30`

Problem G. Easy Problem

Input file:	`maxdiv.in`
Output file:	`maxdiv.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Let $A(x)$ be the number of divisors of x. Consider the following functions.

$$B(x) = \max\{k : k = A(q), q \in N, q \le x\}$$
$$C(x) = \min\{k : A(k) = B(x), k \in N\}$$

Given N you are to find $C(N)$ and $B(N)$.

Input

The input consists of multiple testcases. The first line contains one positive integer T ($T \le 100$) — the number of testcases. The following T lines describe the testcases. The i-th of these lines contains one integer N_i ($1 \le N_i \le 10^{18}$).

Output

For each testcase output the answer on a separate line. The i-th line should contain $C(N_i)$ and $B(N_i)$.

Example

maxdiv.in	maxdiv.out

5
1
10
100
1000
10000
	1 1
	6 4
	60 12
	840 32
	7560 64

Problem H. Periodic Sum

Input file:	`periodicsum.in`
Output file:	`periodicsum.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Let $S(t)$ be the sum of integers represented by all substrings of the decimal representation of t. For example, $S(1205) = 1 + 2 + 0 + 5 + 12 + 20 + 05 + 120 + 205 + 1205 = 1575$. Note that some substrings can have leading zeros. Let $F(t, k)$ be the number which decimal representation is obtained by repeating the decimal representation of t k times. For example, $F(1205, 3) = 120512051205$. Given numbers P, K and M, calculate $S(F(P, K))$ modulo M.

Input

The first line of the input contains one integer P ($1 \le P < 10^{100000}$). The second line contains two integers K and M ($1 \le K, M \le 10^9$).

Output

Output the answer on a single line.

Example

periodicsum.in	periodicsum.out
1205 3 999999999	847123538

Problem I. Polyline

Input file:	`polyline.in`
Output file:	`polyline.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

There is a two-segment polyline L on the plane and points A and B not belonging to L. It is required to determine the infimum of lengths of paths between A and B which do not contain any points of the polyline L.

Input

The first line of the input file contains one positive integer N — the number of test cases ($1 \leq N \leq 5000$). Each of the next N lines contains one test case. Each test case is written as 10 numbers separated by space: $x_A, y_A, x_B, y_B, x_1, y_1, x_2, y_2, x_3, y_3$ where (x_A, y_A) and (x_B, y_B) are the coordinates of the points A and B respectively and (x_1, y_1) – (x_2, y_2) and (x_2, y_2) – (x_3, y_3) are the segments of the polyline. All points within the test case are different. All numbers in test cases are integers and doesn't exceed 10 by absolute value.

Output

For each test case output the answer on a separate line. The answer is accepted if it differs from the jury's answer by less than 10^{-6}.

Example

`polyline.in`	`polyline.out`

`3`
`1 2 5 6 4 4 5 2 1 6`
`2 2 4 3 1 3 3 3 3 1`
`2 1 4 4 3 2 4 3 1 4`
	`8.000000`
	`3.650282`
	`3.828427`

Problem J. Improbability Theory

Input file:	`theory.in`
Output file:	`theory.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Prof. W. Booster, a famous Galactic Institute professor, continues his work on the Ultimate Improbability Theory. Currently, it consists of N hypotheses. Some of them are based on other hypotheses of the Theory. Like any other theory which deals with Improbability effects, Booster's one is extremely complicated so there's nothing surprising about the fact that cause-and-effect relations between the hypotheses can form cycles.

After several desperate, but still rather unsuccessful attempts to explain results of the very last experiment by means of the Theory, the professor began to suspect that something must be definitely wrong with some of the hypotheses.

To check his theory Prof. Booster constructed a special computer which allows to determine whether a hypothesis is *admissible*. The time it takes to check the i-th hypothesis is equal to T_i. The professor also managed to compute the probabilities P_i of the i-th hypothesis being admissible. It's worth to note that the computer allows to check hypotheses in any order but no more than one at a time. Futhermore, the admissibility of any hypothesis doesn't depend in any way on the admissibility of any other hypothesis.

The professor considers a hyphothesis *correct* if it's admissible and all of the hypotheses it is (directly or indirectly) based upon are admissible too. Now Prof. Booster is going to positively identify which of hypotheses are correct and which are not. Help him to calculate the expected time it will take if the checking process would be organized in an optimal way.

Input

The first line contains an integer N ($1 \le N \le 500$) — the number of hypotheses in the Theory. Next N lines describe hypotheses. The i-th of these lines starts from three numbers t_i ($1 \le t_i \le 1000$), p_i ($0 < p_i < 1$), and k_i ($0 \le k_i \le N$), followed by k_i integers — numbers of hypotheses the i-th hypothesis is directly based upon. Hypotheses are numbered by integers from 1 to N.

Output

Output the expected time required to determine the correctness of every hypothesis. The answer must be accurate up to 10^{-3}.

Examples

theory.in	theory.out

4
5 0.3 1 2
6 0.99 0
2 0.2 1 4
2 0.2 0
	13.350

3
1 0.5 1 2
2 0.5 1 3
3 0.5 1 1
	2.750

Problem K. Tree

Input file:	`treedist.in`
Output file:	`treedist.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Consider a tree consisting of N vertices. Distance between vertices u and v is the minimal number of edges in a path connecting u and v. You are to answer the following queries. Given a vertex v_i and distance d_i find some vertex u_i such that distance between v_i and u_i equals to d_i (if such vertex exists).

Input

The first line contains two integers N ($1 \le N \le 20000$) and Q ($1 \le Q \le 50000$) — the number of vertices and the number of queries. Each of the following $N - 1$ lines describes an edge and contains two integers — numbers of vertices connected by the edge. Vertices are numbered from 1 to N. The next Q lines describe the queries. Each query is described by a line containing two numbers v_i ($1 \le v_i \le N$) and d_i ($0 \le d_i \le N$).

Output

The output should consist of Q lines. The i-th line should contain a vertex number u_i — the answer to the i-th query. If there are no required vertices, output 0 instead.

Example

treedist.in	treedist.out

```
9 10
1 8
1 5
1 4
2 7
2 5
3 6
5 9
6 9
5 4
8 1
4 3
2 4
9 3
1 1
5 2
3 5
6 4
7 3
```
	```
	0
	1
	2
	3
	4
	5
	6
	7
	8
	9
	```