Problem A. Abelian Groups

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

Andrew has just made a breakthrough in group theory: he realized that he can classify all finite Abelian groups (not much of a breakthrough, indeed).

Given n, how many Abelian groups with n elements exist up to isomorphism?

To help you solve this problem we provide some definitions and theorems from basic algebra (most are cited from Wikipedia).

An abelian group is a set, A, together with an operation '$\cdot$' that combines any two elements a and b to form another element denoted $a \cdot b$. The symbol '$\cdot$' is a general placeholder for a concretely given operation. To qualify as an abelian group, the set and operation, $(A, \cdot)$, must satisfy five requirements known as the abelian group axioms:

- *Closure*: for all a, b in A, the result of the operation $a \cdot b$ is also in A.
- *Associativity*: for all a, b and c in A, the equation $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ holds.
- *Identity element*: there exists an element e in A, such that for all elements a in A, the equation $e \cdot a = a \cdot e = a$ holds.
- *Inverse element*: for each a in A, there exists an element b in A such that $a \cdot b = b \cdot a = e$, where e is the identity element.
- *Commutativity*: for all a, b in A, $a \cdot b = b \cdot a$.

An example of an abelian group is a *cyclic group* of order n: the set is integers between 0 and $n-1$, and the operation is sum modulo n.

Given two abelian groups G and H, their *direct sum* is a group where each element is a pair (g, h) with g from G and h from H, and operations are performed on each element of the pair independently.

Two groups G and H are *isomorphic* when there exists a one-to-one mapping f from elements of G to elements of H such that $f(a) \cdot f(b) = f(a \cdot b)$ for all a and b.

The *fundamental theorem of finite abelian groups* states that every finite abelian group is isomorphic to a direct sum of several cyclic groups.

The *Chinese remainder theorem* states that when m and n are coprime, a cyclic group of order mn is isomorphic to the direct sum of the cyclic group of order m and the cyclic group of order n.

Input

First and only line of the input file contains an integer n, $1 \le n \le 10^{12}$.

Output

In the only line of the output file write the number of abelian groups with n elements.

Sample input and output

standard input	standard output
5	1
4	2
12	2

Problem B. Coins

Input file:	`standard input`
Output file:	`standard output`
Time limit:	15 seconds
Memory limit:	256 megabytes

Andrew has just made a breakthrough in gambling: he has invented a way to make coins which have probability p of landing heads-up and $1 - p$ of landing tails-up when tossed, where p is some number between 0 and 1. He can't, however, control the number p itself, and creates coins with any p from 0 to 1 with equal probability. That is, the value of p is a random value uniformly distributed on $[0, 1]$.

Andrew has generated two coins independently. One with probability p and another with probability q of landing heads-up. Random values p and q are both uniformly distributed on $[0, 1]$ and are independent. Of course, neither Andrew nor we know the numbers p and q, we can only try to guess them using our observations. The observations are the following: the first coin was tossed n_1 times, and m_1 of them landed heads-up. The second coin was tossed n_2 times, and m_2 of them landed heads-up. Your task is to compute the probability that $p < q$.

Input

The first line of the input file contains one integer T ($1 \le T \le 100\,000$) — the number of test cases to solve. Each of the following T lines contains 4 integers each: n_1, m_1, n_2, m_2. $1 \le n_1, n_2 \le 1000$, $0 \le m_1, m_2 \le 50$, $0 \le m_1 \le n_1, 0 \le m_2 \le n_2$.

Output

For each test case output one line with a floating-point number, the probability of $p < q$. Your answer will be considered correct if it is within 10^{-4} of the right answer.

Sample input and output

standard input	standard output
4 2 1 4 3 8 4 16 8 2 0 6 1 2 0 2 1	0.7142857142 0.5000000000 0.5333333333 0.8000000000

Problem C. Greatest Greatest Common Divisor

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

Andrew has just made a breakthrough in sociology: he realized how to predict whether two persons will be good friends or not. It turns out that each person has an inner *friendship number* (a positive integer). And the *quality of friendship* between two persons is equal to the greatest common divisor of their friendship number.

That means there are *prime* people (with a prime friendship number) who just can't find a good friend, and... Wait, this is irrelevant to this problem.

You are given a list of friendship numbers for several people. Find the highest possible quality of friendship among all pairs of given people.

Input

The first line of the input file contains an integer n ($2 \le n \le 100\,000$) — the number of people to process. The next n lines contain one integer each, between 1 and 1 000 000 (inclusive), the friendship numbers of the given people. All given friendship numbers are distinct.

Output

Output one integer — the highest possible quality of friendship. In other words, output the greatest greatest common divisor among all pairs of given friendship numbers.

Sample input and output

standard input	standard output
4 9 15 25 16	5

Problem D. Circular Island

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

Andrew has just made a breakthrough in geography: he has found an island that was previously unknown to the civilized world. This island has a shape of perfect circle and it is inhabited by two tribes, *Java* and *Seeplusplus*.

After a brief contact with the aborigines, Andrew found out that the boundary between the lands of the tribes is a straight line. Moreover, he knows the locations of several Java villages and of several Seeplusplus villages (each is of course located within or on the boundary of the corresponding tribe's land).

Now he needs to find out what is the minimal and maximal possible area of Java's land. Help him!

Input

The first line of the input file contains one integer r — the radius of the island ($1 \le r \le 10^9$).

The next line contains one integer n ($1 \le n \le 50\,000$) — the number of Java villages. Each of the next n lines contains two integers x and y — the coordinates of Java villages.

The next line contains one integer m ($1 \le m \le 50\,000$) — the number of Seeplusplus villages. Each of the next m lines contains two integers x and y — the coordinates of Seeplusplus villages.

The center of the island has coordinates $(0, 0)$, each village is within the island and at least $r/10$ away from the island boundary. No two villages coincide. The input is guaranteed to be valid — there will always be at least one straight line separating the Java villages from the Seeplusplus villages.

Output

Output two floating-point numbers, separated with a space — the minimal and maximal possible area of the Java land. An area will be considered correct if it is within 10^{-6} relative error of the right answer.

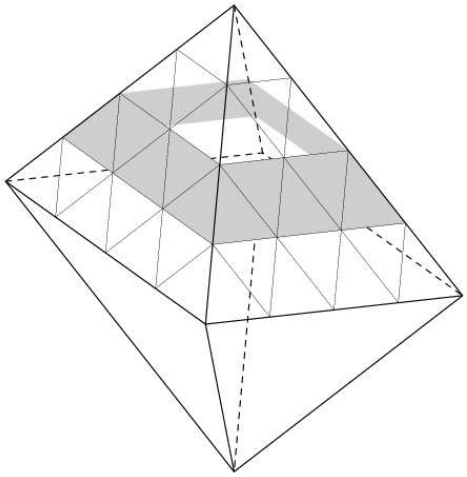
Sample input and output

standard input	standard output
6 2 3 4 -3 4 1 0 0	12.389928320447176 56.548667764616276

Problem E. Octahedron And Dominoes

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

Andrew has just made a breakthrough in puzzle making: he's invented a new puzzle that is both simple to understand and unlike everything that has been invented before.



His new puzzle looks like an octahedron with each of its (triangular) faces split into n^2 smaller triangles by $3(n-1)$ lines, $n-1$ parallel to each side of the face.

Some of smaller triangles are black and some are white. The objective is to cover all white triangles with triangular dominoes without overlapping, leaving all black triangles uncovered. A triangular domino is two small triangles with one common side. Please note that a domino can cover one small triangle on a side of the octahedron and another on the adjacent side of the octahedron (in other words, you can bend your domino in the middle). You must place each domino so that it covers exactly two white small triangles (that means you can't cover some part of a triangle with one domino and the rest with another domino, for example).

How many solutions does the given puzzle have?

Input

The first line of the input file contains an integer n ($1 \le n \le 4$).

The next $2n$ lines describe which small triangles are black and which are white. Imagine the octahedron standing on its vertex, with one of its other vertices pointing on us. Then the small triangles can be split into horizontal layers. The topmost layer contains 4 triangles, one per each side that ends in the top vertex. The next layer contains 12 triangles, 3 per each side that ends in the top vertex, and so on. The size of each layer increases by 8 until it reaches $8n-4$ in the middle, then the next layer is again of $8n-4$ small triangles (here we switched from the top 4 faces to the bottom 4 faces), and then the size of each next layer is less by 8 until we reach the bottommost layer that has just 4 triangles.

The octrahedron is given layer-by-layer, and each layer is started with the leftmost small triangle on the visible part of the octahedron (the four "front" faces), goes from left to right along the visible part, and then switches to the invisible part and goes back from right to left. One layer is highlighted on the picture above.

Each small triangle is specified either by '`.`', denoting a white triangle, or by '`*`', denoting a black triangle.

Output

Output one integer — the number of possible coverings of white triangles of the given octahedron with triangular dominoes.

Sample input and output

standard input	standard output
2 .*** .....*... **********	2
3 .... ********** ..... ********** ..... ****	8

Problem F. Digits Permutation

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

Andrew has just made a breakthrough in the world of number 17: he realized that it's rather easy to permute the digits in almost any given number to get a number divisible by 17.

You are given a positive integer n. You must find a permutation of its digits that is divisible by 17.

Input

Input file contains single integer n, $1 \le n \le 10^{17}$.

Output

Output any permutation of digits of n that is divisible by 17. The output permutation may not start with a zero. If there is no such permutation, output -1.

Sample input and output

standard input	standard output
17	17
2242223	2222342
239	-1

Problem G. Running City

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

Andrew has just made a breakthrough in urban orienteering: he realized that each urban orienteering contest, like the upcoming "Running City Petrozavodsk", can be represented as a simple problem in graph theory. Your task is to get from point A to point B in the city as fast as you can. City map consists of roads (corresponding to edges of a graph) and road intersections (corresponding to nodes of a graph). Points A and B are road intersections, of course. You know the running times for all roads in the city. However, you won't be able to solve your problem with a usual Dijkstra algorithm, because there are some additional difficulties.

There are some *routes* in the city which slow you down. Some friends in the organizing committee gave you the secret map where those routes are marked. A route can be any simple path (no intersection can appear twice) in the graph.

The difficulty is: if you run through the whole route, a group of volunteers waits for you in the end of the route, and the celebration begins... It slows you down by the time equal to the time you spent running through the route. So it's like if you ran through the route 2 times. Also, if your path contains several such routes as subpaths, then each of them counts. For example, there can be two exactly equal routes in the input, and if your path contains such route, then it's actually as if you ran 3 times through the route.

Find the shortest time to get from one node to another in such a strange graph with "bonuses".

Input

First line of the input file contains five integers n, m, r, S, T — number of nodes, edges, special routes in the graph, start node and finish node respectively. $2 \le n \le 1\,000$, $0 \le m \le 10\,000$, $0 \le r \le 1\,000$, $1 \le S, T \le n$, $S \ne T$. The nodes in the graph are numbered from 1 to n. Each of the following m lines contains three integers a, b, c, $(1 \le a, b \le n, a \ne b, 1 \le c \le 1\,000)$, where a and b are nodes and c is the time to run through the edge (a, b). The graph is directed: you can't run from b to a along the edge (a, b). There are no more than 10 edges going out from each node. Each of the next r lines contains a description of a special route. It begins with integer k — the number of edges in the route. Then k integers follow — the numbers of the edges in the route. Edges are numbered from 1 to m in the same order that they are written above. Sum of lengths of all routes is not more than $2m$. Each edge occurs no more than in 10 special routes. Each special route is correct: no vertex can appear twice in any given route, and the end of each edge of the route except the last one coincides with the start of the next edge.

Output

If there is a way from S to T, print the minimal time to get from S to T and any optimal path, otherwise just print -1. The format of output in case a way exists: on the first line print the minimal time, on the second line print the number of edges in the path and on the last line print the sequence of numbers of edges (ranging from 1 to m) that form the optimal path.

Sample input and output

standard input	standard output
3 3 1 1 3 1 2 2 2 3 1 1 3 2 1 3	3 2 1 2
3 3 3 1 3 1 2 2 2 3 2 1 3 1 1 3 1 3 1 3	4 2 1 2
4 3 3 1 4 1 2 3 2 3 2 3 4 1 3 1 2 3 2 2 3 1 3	16 3 1 2 3

Problem H. Square Palindrome

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

Andrew has just made a breakthrough in computer science: he realized how to quickly find the largest palindrome square on a given rectangle of letters. Can you do the same?

A square consisting of n rows of n letters each is a *palindrome square* of size n if each row and each column of this square is a palindrome string. A string is a *palindrome string* if its first letter is the same as its last letter, its second letter is the same as its next-to-last letter, and so on.

Input

The first line of the input file contains two integers h and w ($1 \le h, w \le 700$) — the height and width of the given rectangle of letters. The next h lines contain w lowercase English letters each — the given rectangle of letters itself.

Output

Output the coordinates of the largest palindrome square that is a part of the given rectangle of letters. Output four integers: the first row of the square, the first column of the square, the last row of the square, the last column of the square. The rows are numbered from 1 to h, the columns are numbered from 1 to w. If there are several solutions, output any.

Sample input and output

standard input	standard output
5 10 abccbfghij abccbfghij abccbfghij abccbfghij abcdefghij	1 2 4 5

Problem I. Prefixes and suffixes

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	256 megabytes

Andrew has just made a breakthrough in biology: he realized that most of genome's characteristics are determined by the absence or presence of certain prefixes and suffixes. So he needs to quickly test large arrays of genomes for that.

More formally, we represent the genome as a string of lowercase English letters. We are given several genomes, and need to answer the following type of question: how many genomes from the given set have the given string p as prefix **and** the given string s as suffix.

Input

The first line of the input file contains an integer n — the number of genomes.

The next n lines contain a non-empty string of lowercase English letters each, representing the given genomes. The total length of all given genomes doesn't exceed 100 000.

The next line contains an integer m — the number of questions to answer. The next m lines contain two non-empty strings of lowercase English letters each, the first denoting the prefix and the second denoting the suffix. The prefix and suffix are separated with a single space. The total length of all given prefixes plus the total length of all given suffixes doesn't exceed 200 000.

Output

Output m integers one per line. i-th output line should contain the number of genomes that have i-th given prefix and i-th given suffix.

Sample input and output

standard input	standard output
3 aaaaa abacabaa avtobus 6 a a a aa aa a aaaaa aaaa abac caba abac a	2 2 1 1 0 1

Problem J. Subsequences Of Substrings

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

Andrew has just made a breakthrough in steganography: he realized that one can hide a message in a bigger text by making the message a subsequence of the text.

We remind that a string s is called a *subsequence* of string t if one can remove some (possibly none) letters from t and obtain s.

Andrew has prepared a text (represented by a string) with a hidden message (represented by another string which is a subsequence of the first string). But it turns out that he doesn't have enough space to write the text, so he wonders if he can remove some letters from the beginning and/or the end of his text in such a way that the hidden message still stays a subsequence of it.

You should find out how many ways are there to remove some (possibly none) letters from the beginning of the given text and some (possibly none) letters from the end of the given text in such a way that the given message is a subsequence of the remaining string. Two ways are distinct if the number of letters removed from the beginning or from the end or both are distinct, even if the resulting string is the same.

Input

The first line of the input file contains the text — a non-empty string of lowercase English letters, no more than 100 000 letters long. The second line of the input file contains the message — a non-empty string of lowercase English letters, no more than 100 letters long.

It is guaranteed that the message is a subsequence of the given text.

Output

Output one integer — the sought number of ways.

Sample input and output

standard input	standard output
abraaadabraa baa	23

Problem K. Treediff

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

Andrew has just made a breakthrough in complexity theory: he thinks that he can prove $P = NP$ if he can get a data structure which allows to perform the following operation quickly. Naturally, you should help him complete his brilliant research.

Consider a rooted tree with integers written in the leaves. For each internal (non-leaf) node v of the tree you must compute the minimum absolute difference between all pairs of numbers written in the leaves of the subtree rooted at v.

Input

The first line of the input file contains two integers n and m — overall number of nodes in the tree and number of leaves in the tree respectively. $2 \le n \le 50\,000, 1 \le m < n$. All nodes are numbered from 1 to n. Node number 1 is always the root of the tree. Each of the other nodes has a unique parent in the tree. Each of the next $n - 1$ lines of the input file contains one integer — the number of the parent node for nodes $2, 3, \ldots, n$ respectively. Each of the last m lines of the input file contains one integer ranging from $-1\,000\,000$ to $1\,000\,000$ — the value of the corresponding leaf. Leaves of the tree have numbers from $n - m + 1$ to n.

Output

Output one line with $n - m$ integers: for each internal node of the tree output the minimum absolute difference between pairs of values written in the leaves of its subtree. If there is only one leaf in the subtree of some internal node, output number $2^{31} - 1$ for that node. Output the answers for the nodes in order from node number 1 to $n - m$.

Sample input and output

standard input	standard output
5 4 1 1 1 1 1 4 7 9	2
5 4 1 1 1 1 1 4 7 10	3
7 4 1 2 1 2 3 3 2 10 7 15	3 3 8
2 1 1 100	2147483647