Problem A. Senseless and Merciless

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Yandex has a well-defined subordination structure in the form of a tree with a designated root. Each of the N employees has a corresponding vertex in that tree and an integer identification number v ($0 \le v < N$). The root is a vertex with identifier 0.

Since this organizational tree grows bigger from year to year, Yandex need experts in algorithms and data structures in the field of trees. As a test, all candidates are asked to complete the following senseless and merciless task.

Given a tree of employees, where for each person (except the head of the company), that person's boss is known, you need to sequentially answer M queries. Each query consists of two vertices u and v.

Assume that the shortest path between u and v is $p_1, p_2, \ldots, p_k$, where $p_1 = u$ and $p_k = v$. In order to find the answer for the query (u, v), you are supposed to compute the sum

$$\sum_{w=0}^{N-1} \min\left(d(w, p_1), \ldots, d(w, p_k)\right) \cdot w,$$

where $d(w, p_i)$ is the distance in edges between vertices w and p_i, and—yes, you got it correctly—the $\min(\ldots)$ in the summation is weighted by the vertex identification number itself.

Input

The first line of the input contains a single integer N, the number of vertices in the tree ($2 \le N \le 100\,000$).

The next line contains $N - 1$ integers separated by spaces, where i-th number (counting from 1) is the identifier of parent vertex for vertex i. It is guaranteed that the given graph is a tree rooted at vertex 0.

On the third line, there is a single integer M, the number of queries ($0 \le M \le 100\,000$).

Then M lines follow. Each of them consists of two integers u and v: pairs of vertex identifiers for which the aforementioned sum must be computed.

Output

For each query in the input, write a single integer on a separate line that is the answer to this query.

Example

standard input	standard output
10 0 0 1 1 1 1 1 3 1 3 1 6 5 0 8 7	48 47 28

Problem B. Multithreading

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

As improving of single processing unit performance has been stagnating over last decade, many IT companies have to use multiprocessing approach when there is a need in heavy computations.

Yandex is one of such companies, and here, we are constantly inventing state-of-the-art technologies for solving multithreading and multiprocessing problems. As an example of these technologies, we would like to introduce you our new approach on context switch for multithreading tasks.

More formally, we consider N threads, and for simplicity suppose that each thread is a function which contains a_i atomic instructions (one atomic instruction takes one time slice to execute).

Each time slice, our planner does the following steps:

1. Consider all threads which have not been executed yet.
2. Pick one of these threads at random with equal probability and execute exactly one of its instructions.

   If all instructions of a thread are executed, the whole thread is considered to be executed.

After all instructions of all threads are executed, we have some order in which the threads became executed. For each thread i, we are interested in its expected position e_i in that list.

Formally, the expected position for a thread is calculated as follows. Consider all different orders $p_1, p_2, \ldots, p_{N!}$ in which the threads become executed and the corresponding probabilities $r_1, r_2, \ldots, r_{N!}$ of these orders. The expected place in the list for the thread i is

$$\sum_{j=1}^{N!} p_j(i) r_j,$$

where $p_j(i)$ is the position of thread i in order j.

Input

The first line contains the number of threads N ($1 \le N \le 10^5$). The second line contains N integers $a_1, a_2, \ldots, a_N$ separated by spaces ($1 \le a_i \le 5000$).

Output

Output N lines: i-th line must contain the value e_i with absolute or relative error at most 10^{-6}.

Example

standard input	standard output
2 1 5000	1.0000000000 2.0000000000

Problem C. Polynomial

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

The department of machine learning algorithms optimization at Yandex is actively searching the best universal polynomial representation for all used formulae.

They operate with polynomials of the form $p = a_0 + a_1x + \ldots + a_{n-1}x^{n-1} + a_nx^n$ where $0 \le a_i < 10^9 + 7$. All operations with polynomials, its coefficients and values at points are calculated modulo $10^9 + 7$ to avoid big or non-integer numbers.

Recently, one of our new interns has invented the new short polynomial representation which outperforms current results and seems to be very promising.

Consider a non-zero polynomial $p(x) = a_0 + a_1x + \ldots + a_{n-1}x^{n-1} + a_nx^n$ and its factorization

$$p(x) = p_1(x) \cdot p_2^2(x) \cdot p_3^3(x) \cdot \ldots \cdot p_k^k(x),$$

where $\deg(p_k(x)) \ge 1$ and the factorization has the maximum possible k, then the maximum possible $\deg(p_k(x))$, then the maximum possible $\deg(p_{k-1}(x))$ and so on. Here $\deg(q(x))$ is the degree of the polynomial $q(x)$. All the polynomials $p_1, p_2, \ldots, p_k$ meet the same requirements on coefficients and calculation of values at points as polynomial p.

Unfortunately, the intern hasn't come up with a fast solution for this factorization problem. So, we want to check if you are the one that can implement an efficient solution. And if you can't, we promise to forget about this hopeless new idea of such factorization type.

Your task is to prove us you can.

Input

The first line consists of one integer n, the degree of the input polynomial p ($1 \le n \le 100$).

The second line contains all $n + 1$ integers: the coefficients $a_0, a_1, a_2, \ldots, a_n$ of the polynomial ($0 \le a_i < 10^9 + 7$, $a_n \ne 0$).

Output

In the first line, output a single integer k.

In the second line, output k integers: the degrees $\deg(p_1(x)), \deg(p_2(x)), \ldots, \deg(p_k(x))$.

Examples

standard input	standard output
5 0 0 2 6 6 2	3 0 1 1
2 224999993 70000 1	2 0 1
2 1 1 1	1 2

Problem D. Openspaces

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

Each floor of Yandex office consists of several open spaces lying on the same plane. Employee number i is sitting at a desk, which for the sake of simplicity we assume to be a point with coordinates (x_i, y_i). It is also known that each employee belongs to exactly one team working on something very valuable for Yandex. Of course, there are no employees sharing a desk in the company.

All the members of each team are located in the vertices of some convex polygon. Though the polygons may intersect, they do not meet at vertices. Additionally, no three desks in one team are collinear.

No one remembers why such strange requirements exist, but the Department of Employee Placement works hard to meet them.

One of the regular duties of this department is to count the number of different common outer tangents for each pair of team polygons. A common outer tangent for two polygons is a line that touches the boundary of each polygon at one or more points such that all the inner points of both polygons lie by the same side of it.

Help the Department of Employee Placement to automatize this task if you know that there are no more than 10^5 employees in Yandex.

Please remember that the Department has a lot of other important things to do, so make sure your solution is as fast as possible. We also strongly recommend to test it with 64-bit compilers.

Input

In the first line of the input there is a single integer N, number of different teams in Yandex ($1 \le N \le 400$).

Then N team descriptions follow.

Each description starts with a line with an integer N_i: the number of members in the i-th team ($3 \le N_i \le 10^5$). Then follow N_i pairs of integers, one pair per line. A pair x_{ij}, y_{ij} denotes the planar coordinates of the j-th member of the i-th team ($-10^9 \le x_{ij}, y_{ij} \le 10^9$).

The sum of all N_i is guaranteed to be at most 10^5.

Coordinates of team-members within one team are given in counter-clockwise order.

Output

Output N lines with N space-separated integers on each line.

The j-th number in the i-th ($j \ne i$) line must be equal to the number of common outer tangents of i-th and j-th team.

The i-th number in the i-th line must always be zero.

Example

standard input	standard output
3 4 3 1 8 1 8 6 3 6 3 5 0 13 5 2 7 4 2 5 9 1 12 2 3 8	0 6 4 6 0 6 4 6 0

Problem E. Game

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

All Yandex developers have their favorite way to relax between working hours. Usually they play kicker, table tennis or board games, but Yasha's group has invented their own card game to spend their leisure time.

At the start of the game, there are n cards with some positive integers on the table, and some positive integer M (starting number) is written on the whiteboard. Two players take turns to make their moves. For each move, player picks up a card on the table, wipes the number off the whiteboard and writes down the quotient of integer division of the last number from the board by the number from the card. After the move, the card that has been played is put away from the table. If after a player's turn the number on the whiteboard is 0, he loses.

Yasha is the youngest member of his group, so his turn is always second, and he thinks that it might be unfair. In order to test his hypothesis, he studied the game in detail, and he now knows all numbers $a_1, a_2, \ldots, a_n$ on the cards and the boundaries L and R such that the starting number M is always picked between them.

Help Yasha to count all the numbers M such that $L \le M \le R$ and, if the game started with M written on the board, Yasha would win, provided that he and his opponent played in the optimal way.

Input

The input consists of three lines.

On the first line of the input, there is a single integer n: the number of cards on the table at the start of the game ($2 \le n \le 10$).

On the second line, there are n space-separated integers $a_1, a_2, \ldots, a_n$: the numbers written on the cards ($1 \le a_i \le 1000$).

On the third line, there are two integers L and R: the boundaries for the starting number M ($1 \le L \le R \le 10^{18}$, $R < a_1 \cdot a_2 \cdot \ldots \cdot a_n$).

Output

Output a single integer: the number of winning Ms from $[L, R]$ for Yasha.

Examples

standard input	standard output
3 1 2 1 1 1	1
3 2 3 5 1 10	2

Problem F. Password

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	128 mebibytes

For security reasons, every employee at Yandex must change the password to his or her @yandex-team.ru mailbox once in a while.

Yasha has a remarkable password changing routine.

On his first day at Yandex, he chose the lexicographically smallest sequence of M printable non-whitespace ASCII characters as a password. Each of his next passwords was the lexicographically smallest sequence of M printable non-whitespace ASCII characters that was previously unused by Yasha.

Yasha works at Yandex for so long that he has started to worry that he will run out of sequences of this kind real soon, and decided to change his password choosing strategy in the future.

As for this day, Yasha's password is T.

For his new password, Yasha has come up with a sequence of printable non-whitespace ASCII characters S of length N. This new password is not restricted by T in any way: it can be shorter or longer, or even completely equal to T. Now Yasha wants to estimate the combined similarity between S and all his previous passwords.

The *similarity* of S to a single password P from the past is equal to the length of the longest common substring of S and P. The *combined similarity* to all the passwords starting from the lexicographically smallest one up to his current one is equal to the sum of similarities between S and each of these passwords, taken modulo $10^9 + 7$.

Your task is to help Yasha to compute this combined similarity.

Input

The input consists of two lines.

The first line contains Yasha's future password S of length N ($1 \le N \le 100$).

The second lone contains Yasha's current password T of length M ($1 \le M \le 70$).

Each sequence consists entirely of printable non-whitespace ASCII characters with codes ranging from 33 to 126 inclusive.

Output

Output a single integer that is the answer to this problem.

Example

standard input	standard output
xyz bb	195

Problem G. Random Shuffle Ranking

Input file:	*standard input*
Output file:	*standard output*
Time limit:	9 seconds
Memory limit:	256 mebibytes

Yakov works in Yandex ranking team in the core of the web search itself.

On the eve of an important release, Yakov has noticed that one of the reranking rules is faulty: it rearranges documents in a wrong way. As a conscientious employee, Yakov can not let such a failure affect users' search experience.

But the new binaries are already deployed to production servers, and there is little time left before the faulty functionality will be turned on. So, Yakov decided to resort to desperate measures: he will try to fix everything by changing parameters in server configuration files.

Yakov noticed that after applying the new reranking rules, the documents are shuffled using the `Shuffle` function given below. The default value for `salt` in the configuration file is zero, but Yakov is about to change it.

```
void Shuffle(int a[N], int salt) {
    for (int i = 0; i < N; ++i) {
        int j = i ^ salt; // bitwise xor with salt
        if (i < j) {
            swap(a[i], a[j]);
        }
    }
}
```

You can help Yakov by determining how useful this function is.

Consider a sequence of N integers a_i. Let us denote an array produced by `Shuffle(a, x)` as b_x.

Consider a sequence of arrays b_x for $x = 0, \ldots, N-1$. Your task is to find the sorting permutation p_i ($0 \le p_i < N$) such that the sequence $b_{p_0}, b_{p_1}, \ldots, b_{p_{N-1}}$ is ordered so that b_{p_i} is lexicographically less or equal to $b_{p_{i+1}}$ for $i = 0, \ldots, N-2$. And there is an additional constraint on the sequence p: if for some p_i and p_j, the arrays b_{p_i} and b_{p_j} are equal to each other, then the inequality $p_i < p_j$ holds.

As N in consideration could be large, you are to output the polynomial hash of the sought permutation: $(p_0 \cdot q^{N-1} + p_1 \cdot q^{N-2} + \ldots + p_{N-2} \cdot q + p_{N-1}) \bmod 2^{32}$, where $q = 10^9 + 7$.

Input

The input consists of several test cases. Each test case comprises nine space-separated integers on a single line: N, a_{-3}, a_{-2}, a_{-1}, A, B, C, D and M.

The sequence a_i ($0 \le i < N$) in consideration is generated in the following manner:

$$a_i = (A \cdot a_{i-3} + B \cdot a_{i-2} + C \cdot a_{i-1} + D) \bmod M.$$

The constraints are:

- $N = 2^k$, $0 \le k \le 17$,
- $0 \le a_{-3}, a_{-2}, a_{-1}, A, B, C, D \le 10^9$,
- $1 \le M \le 10^9$,
- the input consists of no more than 2^{17} test cases,
- the sum of all N in a single input does not exceed 2^{21}.

Output

For each test case, output a single integer: the polynomial hash of the sought permutation.

Example

standard input	standard output
4 0 0 1 2 0 3 1 4 4 0 0 1 0 0 1 1 4	1628479554 2008884034

Problem H. Query Matching

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	768 mebibytes

One can tell a lot about a person by the search queries this person issues.

But what is even more characteristic of a person is the way she organizes her search queries into a query session. Even typos she makes and her manner of correcting them matters!

Having acknowledged this fact, Yandex is developing a new dating service with an unprecedented feature: couples are matched basing on their search engine query sessions.

You are given query session logs of two persons. Your task is to compute the score of the match between them.

In the context of this problem, you can assume that a query session is a sequence of keyboard presses by a user. There are only two types of key-presses in each query session. When the user hits a lowercase English letter, this letter is appended to the end of the query in the query input box. When the user hits backspace, the last letter in the query input box (the most recently entered one) disappears. If the user hits backspace when the query input box is empty, nothing happens.

Given a log of key presses in a search query session, one could consider a set of all nonempty strings of lowercase English letters that appeared in the query input box during this search query session. Moreover, one could construct the set of all non-empty substrings of such strings.

Consider two sets of such substrings for two particular users. The score of the match between them is the number of common substrings in these two sets.

Input

Input consists of two search query session logs presented on two separate lines. Each log is a string that consists entirely of lowercase English letters and "<" characters. The "less-than" character denotes a backspace hit by the user.

Each log contains no less than 1 and no more than 100 000 English letters. The total length of each log does not exceed 300 000 characters.

Output

Output a single integer that is the answer to this problem.

Example

standard input	standard output
g<yandex yand<<hoo	10