Codeforces Round #728 (Div. 2)

A. Pretty Permutations

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n cats in a line, labeled from 1 to n, with the i-th cat at position i. They are bored of gyrating in the same spot all day, so they want to reorder themselves such that no cat is in the same place as before. They are also lazy, so they want to minimize the total distance they move. Help them decide what cat should be at each location after the reordering.

For example, if there are 3 cats, this is a valid reordering: $[3, 1, 2]$. No cat is in its original position. The total distance the cats move is $1 + 1 + 2 = 4$ as cat 1 moves one place to the right, cat 2 moves one place to the right, and cat 3 moves two places to the left.

Input

The first line contains a single integer t ($1 \le t \le 100$) — the number of test cases. Then t test cases follow.

The first and only line of each test case contains one integer n ($2 \le n \le 100$) — the number of cats.

It can be proven that under the constraints of the problem, an answer always exist.

Output

Output t answers, one for each test case. Each answer consists of n integers — a permutation with the minimum total distance. If there are multiple answers, print any.

Example

input
2 2 3
output
2 1 3 1 2

Note

For the first test case, there is only one possible permutation that satisfies the conditions: $[2, 1]$.

The second test case was described in the statement. Another possible answer is $[2, 3, 1]$.

B. Pleasant Pairs

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an array $a_1, a_2, \ldots, a_n$ consisting of n **distinct** integers. Count the number of pairs of indices (i, j) such that $i < j$ and $a_i \cdot a_j = i + j$.

Input

The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases. Then t cases follow.

The first line of each test case contains one integer n ($2 \le n \le 10^5$) — the length of array a.

The second line of each test case contains n space separated integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 2 \cdot n$) — the array a. It is guaranteed that all elements are **distinct**.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output the number of pairs of indices (i, j) such that $i < j$ and $a_i \cdot a_j = i + j$.

Example

input
3 2 3 1

```
3
6 1 5
5
3 1 5 9 2
```

output

```
1
1
3
```

Note

For the first test case, the only pair that satisfies the constraints is $(1, 2)$, as $a_1 \cdot a_2 = 1 + 2 = 3$

For the second test case, the only pair that satisfies the constraints is $(2, 3)$.

For the third test case, the pairs that satisfy the constraints are $(1, 2)$, $(1, 5)$, and $(2, 3)$.

C. Great Graphs

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Farmer John has a farm that consists of n pastures connected by one-directional roads. Each road has a weight, representing the time it takes to go from the start to the end of the road. The roads could have negative weight, where the cows go so fast that they go back in time! However, Farmer John guarantees that it is impossible for the cows to get stuck in a time loop, where they can infinitely go back in time by traveling across a sequence of roads. Also, each pair of pastures is connected by at most one road in each direction.

Unfortunately, Farmer John lost the map of the farm. All he remembers is an array d, where d_i is the smallest amount of time it took the cows to reach the i-th pasture from pasture 1 using a sequence of roads. The cost of his farm is the sum of the weights of each of the roads, and Farmer John needs to know the **minimal** cost of a farm that is consistent with his memory.

Input

The first line contains one integer t ($1 \leq t \leq 10^4$) — the number of test cases. Then t cases follow.

The first line of each test case contains a single integer n ($1 \leq n \leq 10^5$) — the number of pastures.

The second line of each test case contains n space separated integers $d_1, d_2, \ldots, d_n$ ($0 \leq d_i \leq 10^9$) — the array d. It is guaranteed that $d_1 = 0$.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, output the minimum possible cost of a farm that is consistent with Farmer John's memory.

Example

input

```
3
3
0 2 3
2
0 1000000000
1
0
```

output

```
-3
0
0
```

Note

In the first test case, you can add roads

- from pasture 1 to pasture 2 with a time of 2,
- from pasture 2 to pasture 3 with a time of 1,
- from pasture 3 to pasture 1 with a time of -3,
- from pasture 3 to pasture 2 with a time of -1,
- from pasture 2 to pasture 1 with a time of -2.

The total cost is $2 + 1 + -3 + -1 + -2 = -3$.
In the second test case, you can add a road from pasture 1 to pasture 2 with cost 1000000000 and a road from pasture 2 to pasture 1 with cost -1000000000. The total cost is $1000000000 + -1000000000 = 0$.

In the third test case, you can't add any roads. The total cost is 0.

D. Tree Array

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a tree consisting of n nodes. You generate an array from the tree by marking nodes one by one.

Initially, when no nodes are marked, a node is equiprobably chosen and marked from the entire tree.

After that, until all nodes are marked, a node is equiprobably chosen and marked from the set of unmarked nodes with at least one edge to a marked node.

It can be shown that the process marks all nodes in the tree.

The final array a is the list of the nodes' labels in order of the time each node was marked.

Find the expected number of inversions in the array that is generated by the tree and the aforementioned process.

The number of inversions in an array a is the number of pairs of indices (i, j) such that $i < j$ and $a_i > a_j$. For example, the array $[4, 1, 3, 2]$ contains 4 inversions: $(1, 2)$, $(1, 3)$, $(1, 4)$, $(3, 4)$.

Input

The first line contains a single integer n ($2 \le n \le 200$) — the number of nodes in the tree.

The next $n - 1$ lines each contains two integers x and y ($1 \le x, y \le n$; $x \ne y$), denoting an edge between node x and y.

It's guaranteed that the given edges form a tree.

Output

Output the expected number of inversions in the generated array modulo $10^9 + 7$.

Formally, let $M = 10^9 + 7$. It can be shown that the answer can be expressed as an irreducible fraction $\frac{p}{q}$, where p and q are integers and $q \not\equiv 0 \pmod{M}$. Output the integer equal to $p \cdot q^{-1} \bmod M$. In other words, output such an integer x that $0 \le x < M$ and $x \cdot q \equiv p \pmod{M}$.

Examples

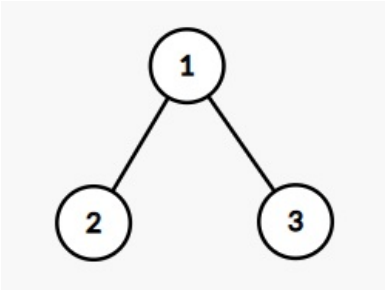
input
3 1 2 1 3
output
166666669

input
6 2 1 2 3 6 1 1 4 2 5
output
500000009

input
5 1 2 1 3 1 4 2 5
output
500000007

Note

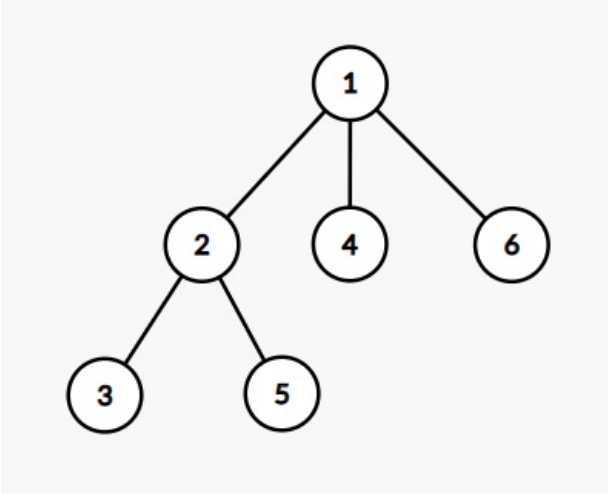
This is the tree from the first sample:



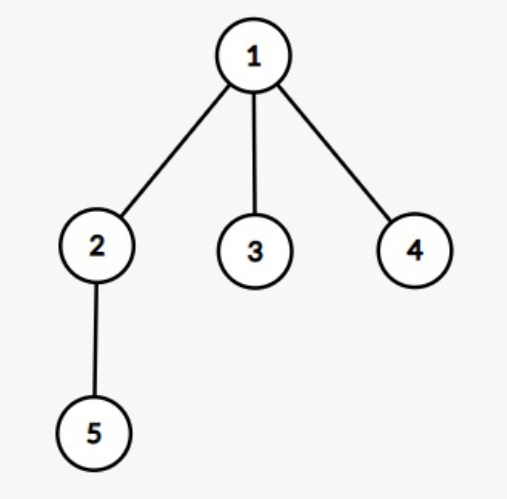
For the first sample, the arrays are almost fixed. If node 2 is chosen initially, then the only possible array is $[2, 1, 3]$ (1 inversion). If node 3 is chosen initially, then the only possible array is $[3, 1, 2]$ (2 inversions). If node 1 is chosen initially, the arrays $[1, 2, 3]$ (0 inversions) and $[1, 3, 2]$ (1 inversion) are the only possibilities and equiprobable. In total, the expected number of inversions is $\frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 2 + \frac{1}{3} \cdot (\frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 1) = \frac{7}{6}$.

$166666669 \cdot 6 = 7 \pmod{10^9 + 7}$, so the answer is 166666669.

This is the tree from the second sample:



This is the tree from the third sample:



E1. Converging Array (Easy Version)

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is the easy version of the problem. The only difference is that in this version $q = 1$. You can make hacks only if both versions of the problem are solved.

There is a process that takes place on arrays a and b of length n and length $n - 1$ respectively.

The process is an infinite sequence of operations. Each operation is as follows:

- First, choose a random integer i ($1 \le i \le n - 1$).
- Then, simultaneously set $a_i = \min\left(a_i, \frac{a_i + a_{i+1} - b_i}{2}\right)$ and $a_{i+1} = \max\left(a_{i+1}, \frac{a_i + a_{i+1} + b_i}{2}\right)$ without any rounding (so values may become non-integer).

See notes for an example of an operation.
It can be proven that array a converges, i. e. for each i there exists a limit a_i converges to. Let function $F(a, b)$ return the value a_1 converges to after a process on a and b.

You are given array b, but not array a. However, you are given a third array c. Array a is good if it contains only **integers** and satisfies $0 \le a_i \le c_i$ for $1 \le i \le n$.

Your task is to count the number of good arrays a where $F(a, b) \ge x$ for q values of x. Since the number of arrays can be very large, print it modulo $10^9 + 7$.

Input

The first line contains a single integer n ($2 \le n \le 100$).

The second line contains n integers $c_1, c_2 \ldots, c_n$ ($0 \le c_i \le 100$).

The third line contains $n-1$ integers $b_1, b_2, \ldots, b_{n-1}$ ($0 \le b_i \le 100$).

The fourth line contains a single integer q ($q = 1$).

The fifth line contains q space separated integers $x_1, x_2, \ldots, x_q$ ($-10^5 \le x_i \le 10^5$).

Output
Output q integers, where the i-th integer is the answer to the i-th query, i. e. the number of good arrays a where $F(a, b) \ge x_i$ modulo $10^9 + 7$.

Example

input
3 2 3 4 2 1 1 -1
output
56

Note
The following explanation assumes $b = [2, 1]$ and $c = [2, 3, 4]$ (as in the sample).

Examples of arrays a that are **not** good:

- $a = [3, 2, 3]$ is not good because $a_1 > c_1$;
- $a = [0, -1, 3]$ is not good because $a_2 < 0$.

One possible good array a is $[0, 2, 4]$. We can show that no operation has any effect on this array, so $F(a, b) = a_1 = 0$.

Another possible good array a is $[0, 1, 4]$. In a single operation with $i = 1$, we set $a_1 = \min(\frac{0+1-2}{2}, 0)$ and $a_2 = \max(\frac{0+1+2}{2}, 1)$. So, after a single operation with $i = 1$, a becomes equal to $[-\frac{1}{2}, \frac{3}{2}, 4]$. We can show that no operation has any effect on this array, so $F(a, b) = -\frac{1}{2}$.

E2. Converging Array (Hard Version)

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is the hard version of the problem. The only difference is that in this version $1 \le q \le 10^5$. You can make hacks only if both versions of the problem are solved.

There is a process that takes place on arrays a and b of length n and length $n-1$ respectively.

The process is an infinite sequence of operations. Each operation is as follows:

- First, choose a random integer i ($1 \le i \le n-1$).
- Then, simultaneously set $a_i = \min\left(a_i, \frac{a_i + a_{i+1} - b_i}{2}\right)$ and $a_{i+1} = \max\left(a_{i+1}, \frac{a_i + a_{i+1} + b_i}{2}\right)$ without any rounding (so values may become non-integer).

See notes for an example of an operation.
It can be proven that array a converges, i. e. for each i there exists a limit a_i converges to. Let function $F(a, b)$ return the value a_1 converges to after a process on a and b.

You are given array b, but not array a. However, you are given a third array c. Array a is good if it contains only **integers** and satisfies $0 \le a_i \le c_i$ for $1 \le i \le n$.

Your task is to count the number of good arrays a where $F(a, b) \ge x$ for q values of x. Since the number of arrays can be very large, print it modulo $10^9 + 7$.

Input
The first line contains a single integer n ($2 \le n \le 100$).

The second line contains n integers $c_1, c_2 \ldots, c_n$ ($0 \le c_i \le 100$).

The third line contains $n-1$ integers $b_1, b_2, \ldots, b_{n-1}$ ($0 \le b_i \le 100$).

The fourth line contains a single integer q ($1 \le q \le 10^5$).

The fifth line contains q space separated integers $x_1, x_2, \ldots, x_q$ ($-10^5 \le x_i \le 10^5$).

Output

Output q integers, where the i-th integer is the answer to the i-th query, i. e. the number of good arrays a where $F(a, b) \ge x_i$ modulo $10^9 + 7$.

Example

input
3 2 3 4 2 1 5 -1 0 1 -100000 100000
output
56 28 4 60 0

Note

The following explanation assumes $b = [2, 1]$ and $c = [2, 3, 4]$ (as in the sample).

Examples of arrays a that are **not** good:

- $a = [3, 2, 3]$ is not good because $a_1 > c_1$;
- $a = [0, -1, 3]$ is not good because $a_2 < 0$.

One possible good array a is $[0, 2, 4]$. We can show that no operation has any effect on this array, so $F(a, b) = a_1 = 0$.

Another possible good array a is $[0, 1, 4]$. In a single operation with $i = 1$, we set $a_1 = \min(\frac{0+1-2}{2}, 0)$ and $a_2 = \max(\frac{0+1+2}{2}, 1)$. So, after a single operation with $i = 1$, a becomes equal to $[-\frac{1}{2}, \frac{3}{2}, 4]$. We can show that no operation has any effect on this array, so $F(a, b) = -\frac{1}{2}$.