Educational Codeforces Round 141 (Rated for Div. 2)

A. Make it Beautiful

3 seconds, 512 megabytes

An array a is called *ugly* if it contains **at least one** element which is equal to the **sum of all elements before it**. If the array is not ugly, it is *beautiful*.

For example:

- the array $[6, 3, 9, 6]$ is ugly: the element 9 is equal to $6 + 3$;
- the array $[5, 5, 7]$ is ugly: the element 5 (the second one) is equal to 5;
- the array $[8, 4, 10, 14]$ is beautiful: $8 \neq 0$, $4 \neq 8$, $10 \neq 8 + 4$, $14 \neq 8 + 4 + 10$, so there is no element which is equal to the sum of all elements before it.

You are given an array a such that $1 \le a_1 \le a_2 \le \cdots \le a_n \le 100$. You have to **reorder** the elements of a in such a way that the resulting array is beautiful. Note that you are not allowed to insert new elements or erase existing ones, you can only change the order of elements of a. You are allowed to keep the array a unchanged, if it is beautiful.

Input

The first line contains one integer t ($1 \le t \le 2000$) — the number of test cases.

Each test case consists of two lines. The first line contains one integer n ($2 \le n \le 50$). The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_1 \le a_2 \le \cdots \le a_n \le 100$).

Output

For each test case, print the answer as follows:

- if it is impossible to reorder the elements of a in such a way that it becomes beautiful, print `NO`;
- otherwise, in the first line, print `YES`. In the second line, print n integers — any beautiful array which can be obtained from a by reordering its elements. If there are multiple such arrays, print any of them.

input

```
4
4
3 3 6 6
2
10 10
5
1 2 3 4 5
3
1 4 4
```

output

```
YES
3 6 3 6
NO
YES
2 4 1 5 3
YES
1 4 4
```

B. Matrix of Differences

2 seconds, 256 megabytes

For a square matrix of integers of size $n \times n$, let's define its **beauty** as follows: for each pair of side-adjacent elements x and y, write out the number $|x - y|$, and then find the number of different numbers among them.

For example, for the matrix $\begin{pmatrix} 1 & 3 \\ 4 & 2 \end{pmatrix}$ the numbers we consider are $|1 - 3| = 2$, $|1 - 4| = 3$, $|3 - 2| = 1$ and $|4 - 2| = 2$; there are 3 different numbers among them (2, 3 and 1), which means that its beauty is equal to 3.

You are given an integer n. You have to find a matrix of size $n \times n$, where each integer from 1 to n^2 occurs exactly once, such that its **beauty** is the maximum possible among all such matrices.

Input

The first line contains a single integer t ($1 \le t \le 49$) – the number of test cases.

The first (and only) line of each test case contains a single integer n ($2 \le n \le 50$).

Output

For each test case, print n rows of n integers — a matrix of integers of size $n \times n$, where each number from 1 to n^2 occurs exactly once, such that its beauty is the maximum possible among all such matrices. If there are multiple answers, print any of them.

input

```
2
2
3
```

output

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

C. Yet Another Tournament

2 seconds, 256 megabytes

You are participating in Yet Another Tournament. There are $n + 1$ participants: you and n other opponents, numbered from 1 to n.

Each two participants will play against each other exactly once. If the opponent i plays against the opponent j, he wins if and only if $i > j$.

When the opponent i plays against you, everything becomes a little bit complicated. In order to get a win against opponent i, you need to prepare for the match for at least a_i minutes — otherwise, you lose to that opponent.

You have m minutes in total to prepare for matches, but you can prepare for only one match at one moment. In other words, if you want to win against opponents $p_1, p_2, \ldots, p_k$, you need to spend $a_{p_1} + a_{p_2} + \cdots + a_{p_k}$ minutes for preparation — and if this number is greater than m, you cannot achieve a win against all of these opponents at the same time.

The final place of each contestant is equal to the number of contestants with strictly more wins $+ 1$. For example, if 3 contestants have 5 wins each, 1 contestant has 3 wins and 2 contestants have 1 win each, then the first 3 participants will get the 1-st place, the fourth one gets the 4-th place and two last ones get the 5-th place.

Calculate the minimum possible place (lower is better) you can achieve if you can't prepare for the matches more than m minutes in total.

Input

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

The first line of each test case contains two integers n and m ($1 \le n \le 5 \cdot 10^5$; $0 \le m \le \sum\limits_{i=1}^{n} a_i$) — the number of your opponents and the total time you have for preparation.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 1000$), where a_i is the time you need to prepare in order to win against the i-th opponent.

It's guaranteed that the total sum of n over all test cases doesn't exceed $5 \cdot 10^5$.

Output

For each test case, print the minimum possible place you can take if you can prepare for the matches no more than m minutes in total.

input
5 4 401 100 100 200 1 3 2 1 2 3 5 0 1 1 1 1 1 4 0 0 1 1 1 4 4 1 2 2 1

output
1 2 6 4 1

In the first test case, you can prepare to all opponents, so you'll win 4 games and get the 1-st place, since all your opponents win no more than 3 games.

In the second test case, you can prepare against the second opponent and win. As a result, you'll have 1 win, opponent 1 — 1 win, opponent 2 — 1 win, opponent 3 — 3 wins. So, opponent 3 will take the 1-st place, and all other participants, including you, get the 2-nd place.

In the third test case, you have no time to prepare at all, so you'll lose all games. Since each opponent has at least 1 win, you'll take the last place (place 6).

In the fourth test case, you have no time to prepare, but you can still win against the first opponent. As a result, opponent 1 has no wins, you have 1 win and all others have at least 2 wins. So your place is 4.

D. Different Arrays

2 seconds, 512 megabytes

You are given an array a consisting of n integers.

You **have to** perform the sequence of $n - 2$ operations on this array:

- during the first operation, you either add a_2 to a_1 and subtract a_2 from a_3, or add a_2 to a_3 and subtract a_2 from a_1;
- during the second operation, you either add a_3 to a_2 and subtract a_3 from a_4, or add a_3 to a_4 and subtract a_3 from a_2;
- ...
- during the last operation, you either add a_{n-1} to a_{n-2} and subtract a_{n-1} from a_n, or add a_{n-1} to a_n and subtract a_{n-1} from a_{n-2}.

So, during the i-th operation, you add the value of a_{i+1} to one of its neighbors, and subtract it from the other neighbor.

For example, if you have the array $[1, 2, 3, 4, 5]$, one of the possible sequences of operations is:

- subtract 2 from a_3 and add it to a_1, so the array becomes $[3, 2, 1, 4, 5]$;
- subtract 1 from a_2 and add it to a_4, so the array becomes $[3, 1, 1, 5, 5]$;
- subtract 5 from a_3 and add it to a_5, so the array becomes $[3, 1, -4, 5, 10]$.

So, the resulting array is $[3, 1, -4, 5, 10]$.

An array is *reachable* if it can be obtained by performing the aforementioned sequence of operations on a. You have to calculate the number of reachable arrays, and print it modulo 998244353.

Input

The first line contains one integer n ($3 \le n \le 300$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 300$).

Output

Print one integer — the number of reachable arrays. Since the answer can be very large, print its remainder modulo 998244353.

input
4 1 1 1 1

output
3

input
5 1 2 3 5 0

output
7

E. Game of the Year

2 seconds, 256 megabytes

Monocarp and Polycarp are playing a computer game. This game features n bosses for the playing to kill, numbered from 1 to n.

They will fight **each** boss the following way:

- Monocarp makes k attempts to kill the boss;
- Polycarp makes k attempts to kill the boss;
- Monocarp makes k attempts to kill the boss;
- Polycarp makes k attempts to kill the boss;
- ...

Monocarp kills the i-th boss on **his** a_i-th attempt. Polycarp kills the i-th boss on **his** b_i-th attempt. After one of them kills the i-th boss, they move on to the $(i + 1)$-st boss. The attempt counters reset for both of them. Once one of them kills the n-th boss, the game ends.

Find all values of k from 1 to n such that Monocarp kills **all bosses**.

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of testcases.

The first line of each testcase contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the number of bosses.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the index of attempt Monocarp kills each boss on.

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le n$) — the index of attempt Polycarp kills each boss on.

The sum of n over all testcases doesn't exceed $2 \cdot 10^5$.

Output

For each testcase, print two lines. The first line should contain a single integer cnt — the number of values of k from 1 to n such that Monocarp kills **all bosses**. The second line should contain cnt distinct integers — the values of k themselves.

input

```
3
3
1 1 1
2 3 1
1
1
1
4
1 4 3 2
3 3 4 1
```

output

```
3
1 2 3
1
1
2
2 4
```

Consider the last testcase of the example.

Let $k = 1$. First, Monocarp makes one attempt to kill the first boss. It's successful, since $a_1 = 1$. Then, Monocarp makes one attempt to kill the second boss. It's unsuccessful, since $a_2 > 1$. So, Polycarp makes an attempt then. It's also unsuccessful, since $b_2 > 1$. Then, Monocarp makes another attempt. It's still unsuccessful, since $a_2 > 2$. This goes on until Polycarp finally kills the boss on his third attempt. Monocarp didn't kill this boss, thus, $k = 1$ isn't the answer.

Let $k = 2$. Monocarp still kills the first boss on his first attempt. Then, he makes two unsuccessful attempts for the second boss. Then, Polycarp makes two unsuccessful attempts. Then, Monocarp makes two more attempts and kills the boss on his fourth attempt. The third boss is similar. First, two unsuccessful attempts by Monocarp. Then, two unsuccessful attempts by Polycarp. Then, Monocarp has two more attempts, but even his first one is successful, since $a_3 = 3$. The fourth boss is also killed by Monocarp. Thus, $k = 2$ is the answer.

F. Double Sort II

2 seconds, 512 megabytes

You are given two permutations a and b, both of size n. A permutation of size n is an array of n elements, where each integer from 1 to n appears exactly once. The elements in each permutation are indexed from 1 to n.

You can perform the following operation any number of times:

- choose an integer i from 1 to n;
- let x be the integer such that $a_x = i$. Swap a_i with a_x;
- let y be the integer such that $b_y = i$. Swap b_i with b_y.

Your goal is to make both permutations **sorted in ascending order** (i. e. the conditions $a_1 < a_2 < \cdots < a_n$ and $b_1 < b_2 < \cdots < b_n$ must be satisfied) using **minimum number of operations**. Note that both permutations must be sorted after you perform the sequence of operations you have chosen.

Input

The first line contains one integer n ($2 \le n \le 3000$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$; all a_i are distinct).

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le n$; all b_i are distinct).

Output

First, print one integer k ($0 \le k \le 2n$) — the minimum number of operations required to sort both permutations. Note that it can be shown that $2n$ operations are always enough.

Then, print k integers $op_1, op_2, \ldots, op_k$ ($1 \le op_j \le n$), where op_j is the value of i you choose during the j-th operation.

If there are multiple answers, print any of them.

input

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

output

```
1
2
```

input

```
2
1 2
1 2
```

output

```
0
```

input

```
4
1 3 4 2
4 3 2 1
```

output

```
2
3 4
```

G. Weighed Tree Radius

6 seconds, 512 megabytes

You are given a tree of n vertices and $n - 1$ edges. The i-th vertex has an initial weight a_i.

Let the *distance* $d_v(u)$ from vertex v to vertex u be the number of edges on the path from v to u. Note that $d_v(u) = d_u(v)$ and $d_v(v) = 0$.

Let the *weighted* distance $w_v(u)$ from v to u be $w_v(u) = d_v(u) + a_u$. Note that $w_v(v) = a_v$ and $w_v(u) \neq w_u(v)$ if $a_u \neq a_v$.

Analogically to usual distance, let's define the *eccentricity* $e(v)$ of vertex v as the greatest weighted distance from v to any other vertex (including v itself), or $e(v) = \max\limits_{1 \le u \le n} w_v(u)$.

Finally, let's define the *radius* r of the tree as the minimum eccentricity of any vertex, or $r = \min\limits_{1 \le v \le n} e(v)$.

You need to perform m queries of the following form:

- v_j x_j — assign $a_{v_j} = x_j$.

After performing each query, print the radius r of the current tree.

Input

The first line contains the single integer n ($2 \le n \le 2 \cdot 10^5$) — the number of vertices in the tree.

The second line contains n integers $a_1, \ldots, a_n$ ($0 \le a_i \le 10^6$) — the initial weights of vertices.

Next $n - 1$ lines contain edges of tree. The i-th line contains two integers u_i and v_i ($1 \le u_i, v_i \le n$; $u_i \neq v_i$) — the corresponding edge. The given edges form a tree.

The next line contains the single integer m ($1 \le m \le 10^5$) — the number of queries.

Next m lines contain queries — one query per line. The j-th query contains two integers v_j and x_j ($1 \le v_j \le n$; $0 \le x_j \le 10^6$) — a vertex and it's new weight.

Output

Print m integers — the radius r of the tree after performing each query.

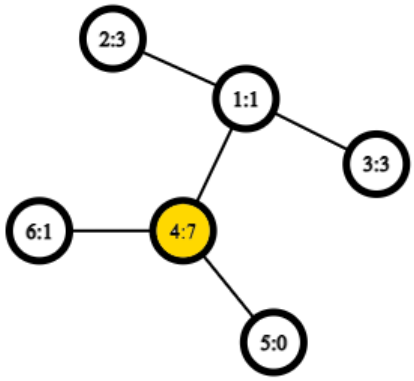
input

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

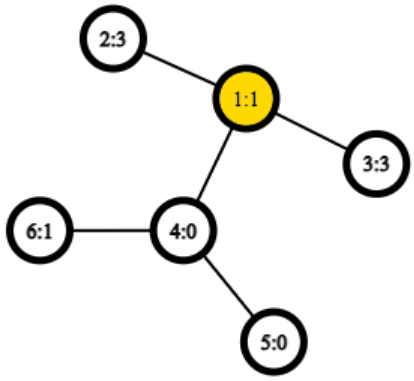
output

```
7
4
5
10
7
```

After the first query, you have the following tree:



The marked vertex in the picture is the vertex with minimum $e(v)$, or $r = e(4) = 7$. The eccentricities of the other vertices are the following: $e(1) = 8, e(2) = 9, e(3) = 9, e(5) = 8, e(6) = 8$.
The tree after the second query:



The radius $r = e(1) = 4$.
After the third query, the radius $r = e(2) = 5$: