A. Tricky Template

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

You are given an integer n and three strings a, b, c, each consisting of n lowercase Latin letters.

Let a template be a string t consisting of n lowercase and/or uppercase Latin letters. The string s matches the template t if the following conditions hold for all i from 1 to n:

- if the i-th letter of the template is **lowercase**, then s_i must be **the same** as t_i;
- if the i-th letter of the template is **uppercase**, then s_i must be **different** from the **lowercase version** of t_i. For example, if there is a letter 'A' in the template, you cannot use the letter 'a' in the corresponding position of the string.

Accordingly, the string doesn't match the template if the condition doesn't hold for at least one i.

Determine whether there exists a template t such that the strings a and b match it, while the string c does not.

Input

The first line contains an integer t ($1 \le t \le 1000$) — the number of test cases.

The first line of each test case contains an integer n ($1 \le n \le 20$) — the length of the given strings.

The next three lines contain the strings a, b and c. Each string consists of exactly n lowercase Latin letters.

Output

For each testcase, print "YES" if there exists a template t such that the strings a and b match it, while the string c does not. Otherwise, print "NO".

Standard Input	Standard Output
4 1 a b c 2 aa bb aa 10 mathforces luckforces adhoccoder 3 acc abd abc	YES NO YES NO

Note

In the first test case, you can use the template "`C`". The first letter of strings a and b differ from '`c`', so they match the template. The first letter of string c equals '`c`', so it doesn't match.

In the third test case, you can use the template "`CODEforces`".

,

B. Forming Triangles

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

You have n sticks, numbered from 1 to n. The length of the i-th stick is 2^{a_i}.

You want to choose **exactly** 3 sticks out of the given n sticks, and form a **non-degenerate** triangle out of them, using the sticks as the sides of the triangle. A triangle is called non-degenerate if its area is **strictly** greater than 0.

You have to calculate the number of ways to choose exactly 3 sticks so that a triangle can be formed out of them. Note that the order of choosing sticks does not matter (for example, choosing the 1-st, 2-nd and 4-th stick is the same as choosing the 2-nd, 4-th and 1-st stick).

Input

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

Each test case consists of two lines:

- the first line contains one integer n ($1 \le n \le 3 \cdot 10^5$);
- the second line contains n integers $a_1, a_2, \dots, a_n$ ($0 \le a_i \le n$).

Additional constraint on the input: the sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, print one integer — the number of ways to choose exactly 3 sticks so that a triangle can be formed out of them.

Standard Input	Standard Output

4
7
1 1 1 1 1 1 1
4
3 2 1 3
3
1 2 3
1
1
	35
	2
	0
	0

Note

In the first test case of the example, any three sticks out of the given 7 can be chosen.

In the second test case of the example, you can choose the 1-st, 2-nd and 4-th stick, or the 1-st, 3-rd and 4-th stick.

In the third test case of the example, you cannot form a triangle out of the given sticks with lengths 2, 4 and 8.

,

C. Closest Cities

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

There are n cities located on the number line, the i-th city is in the point a_i. The coordinates of the cities are given in ascending order, so $a_1 < a_2 < \dots < a_n$.

The distance between two cities x and y is equal to $|a_x - a_y|$.

For each city i, let's define the **closest** city j as the city such that the distance between i and j is not greater than the distance between i and each other city k. For example, if the cities are located in points $[0, 8, 12, 15, 20]$, then:

- the closest city to the city 1 is the city 2;
- the closest city to the city 2 is the city 3;
- the closest city to the city 3 is the city 4;
- the closest city to the city 4 is the city 3;
- the closest city to the city 5 is the city 4.

The cities are located in such a way that for every city, the closest city is unique. For example, it is impossible for the cities to be situated in points $[1, 2, 3]$, since this would mean that the city 2 has two closest cities (1 and 3, both having distance 1).

You can travel between cities. Suppose you are currently in the city x. Then you can perform one of the following actions:

- travel to any other city y, paying $|a_x - a_y|$ coins;
- travel to the city which is the closest to x, paying 1 coin.

You are given m queries. In each query, you will be given two cities, and you have to calculate the minimum number of coins you have to spend to travel from one city to the other city.

Input

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

Each test case is given in the following format:

- the first line contains one integer n ($2 \le n \le 10^5$);
- the second line contains n integers $a_1, a_2, \dots, a_n$ ($0 \le a_1 < a_2 < \dots < a_n \le 10^9$);
- the third line contains one integer m ($1 \le m \le 10^5$);
- then m lines follow; the i-th of them contains two integers x_i and y_i ($1 \le x_i, y_i \le n$; $x_i \ne y_i$), denoting that in the i-th query, you have to calculate the minimum number of coins you have to spend to travel from the city x_i to the city y_i.

Additional constraints on the input:

- in every test case, for each city, the closest city is determined uniquely;
- the sum of n over all test cases does not exceed 10^5;
- the sum of m over all test cases does not exceed 10^5.

Output

For each query, print one integer — the minimum number of coins you have to spend.

Standard Input	Standard Output
1	3
5	8
0 8 12 15 20	1
5	4
1 4	14
1 5	
3 4	
3 2	
5 1	

Note

Let's consider the first two queries in the example from the statement:

- in the first query, you are initially in the city 1. You can travel to the closest city (which is the city 2), paying 1 coin. Then you travel to the closest city (which is the city 3) again, paying 1 coin. Then you travel to the closest city (which is the city 4) again, paying 1 coin. In total, you spend 3 coins to get from the city 1 to the city 4;
- in the second query, you can use the same way to get from the city 1 to the city 4, and then spend 5 coins to travel from the city 4 to the city 5.

,

D. Berserk Monsters

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

Monocarp is playing a computer game (yet again). Guess what is he doing? That's right, killing monsters.

There are n monsters in a row, numbered from 1 to n. The i-th monster has two parameters: attack value equal to a_i and defense value equal to d_i. In order to kill these monsters, Monocarp put a berserk spell on them, so they're attacking each other instead of Monocarp's character.

The fight consists of n rounds. Every round, the following happens:

- first, every alive monster i deals a_i damage to the **closest** alive monster to the left (if it exists) and the **closest** alive monster to the right (if it exists);
- then, every alive monster j which received more than d_j damage **during this round** dies. I. e. the j-th monster dies if and only if its defense value d_j is **strictly less** than the total damage it received **this round**.

For each round, calculate the number of monsters that will die during that round.

Input

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

Each test case consists of three lines:

- the first line contains one integer n ($1 \le n \le 3 \cdot 10^5$);
- the second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$);
- the third line contains n integers $d_1, d_2, \dots, d_n$ ($1 \le d_i \le 10^9$).

Additional constraint on the input: the sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, print n integers. The i-th integer should be equal to the number of monsters that die during the i-th round.

Standard Input	Standard Output

3
5
3 4 7 5 10
4 9 1 18 1
2
2 1
1 3
4
1 1 2 4
3 3 4 2
	3 1 0 0 0
	0 0
	1 1 1 0

Note

Explanation for the first test case of the example:

During the first round, the following happens:

- the monster 1 deals 3 damage to the monster 2;
- the monster 2 deals 4 damage to the monster 1 and the monster 3;
- the monster 3 deals 7 damage to the monster 2 and the monster 4;
- the monster 4 deals 5 damage to the monster 3 and the monster 5;
- the monster 5 deals 10 damage to the monster 4;
- the monster 1 does not die, since it received 4 damage and its defense is 4;
- the monster 2 dies, since it received 10 damage and its defense is 9;
- the monster 3 dies, since it received 9 damage and its defense is 1;
- the monster 4 does not die, since it received 17 damage and its defense is 18;
- the monster 5 dies, since it received 5 damage and its defense is 1.

After the first round, the monsters 1 and 4 stay alive.

During the second round, the following happens:

- the monster 1 deals 3 damage to the monster 4;
- the monster 4 deals 5 damage to the monster 1;
- the monster 1 dies, since it received 5 damage and its defense is 4;
- the monster 4 does not die, since it received 3 damage and its defense is 18.

During the next three rounds, only the 4-th monster is alive, so nothing happens.

,

E. Increasing Subsequences

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

Let's recall that an increasing subsequence of the array a is a sequence that can be obtained from it by removing some elements without changing the order of the remaining elements, and the remaining elements are strictly increasing (i. e $a_{b_1} < a_{b_2} < \dots < a_{b_k}$ and $b_1 < b_2 < \dots < b_k$). Note that an empty subsequence is also increasing.

You are given a positive integer X. Your task is to find an array of integers of length **at most** 200, such that it has exactly X increasing subsequences, or report that there is no such array. If there are several answers, you can print any of them.

If two subsequences consist of the same elements, but correspond to different positions in the array, they are considered different (for example, the array $[2, 2]$ has two different subsequences equal to $[2]$).

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases.

The only line of each test case contains a single integer X ($2 \le X \le 10^{18}$).

Output

For each query, print the answer to it. If it is impossible to find the required array, print `-1` on the first line. Otherwise, print a positive integer n on the first line — the length of the array. On the second line, print n integers — the required array itself. If there are several answers, you can print any of them. All elements of the array should be in the range $[-10^9; 10^9]$.

Standard Input	Standard Output

4
2
5
13
37
	1
	0
	3
	0 1 0
	5
	2 2 3 4 2
	7
	-1 -1 0 0 2 3 -1

,

F. Replace on Segment

```
Input file:     standard input
Output file:    standard output
Time limit:     3 seconds
Memory limit:   256 megabytes
```

You are given an array $a_1, a_2, \dots, a_n$, where each element is an integer from 1 to x.

You can perform the following operation with it any number of times:

- choose three integers l, r and k such that $1 \le l \le r \le n$, $1 \le k \le x$ and **each** element a_i such that $l \le i \le r$ is different from k. Then, for each $i \in [l, r]$, replace a_i with k.

In other words, you choose a subsegment of the array and an integer from 1 to x which does not appear in that subsegment, and replace every element in the subsegment with that chosen integer.

Your goal is to make all elements in the array equal. What is the minimum number of operations that you have to perform?

Input

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

Each test case consists of two lines:

- the first line contains two integers n and x ($1 \le x \le n \le 100$);
- the second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le x$).

Additional constraint on the input: the sum of n over all test cases does not exceed 500.

Output

For each test case, print one integer — the minimum number of operations you have to perform.

Standard Input	Standard Output
3	1
3 2	2
1 2 1	2
6 3	
1 2 3 1 2 3	
12 3	
3 1 3 1 2 1 1 2 3 1 1 3	