Codeforces Round #788 (Div. 2)

A. Prof. Slim

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

One day Prof. Slim decided to leave the kingdom of the GUC to join the kingdom of the GIU. He was given an easy online assessment to solve before joining the GIU. Citizens of the GUC were ~~happy~~ sad to see the prof leaving, so they decided to hack into the system and change the online assessment into a harder one so that he stays at the GUC. After a long argument, they decided to change it into the following problem.

Given an array of n integers $a_1, a_2, \ldots, a_n$, **where** $a_i \neq 0$, check if you can make this array sorted by using the following operation any number of times (possibly zero). An array is sorted if its elements are arranged in a non-decreasing order.

1. select two indices i and j ($1 \leq i, j \leq n$) such that a_i and a_j have **different signs**. In other words, one must be positive and one must be negative.
2. swap the **signs** of a_i and a_j. For example if you select $a_i = 3$ and $a_j = -2$, then they will change to $a_i = -3$ and $a_j = 2$.

Prof. Slim saw that the problem is still too easy and isn't worth his time, so he decided to give it to you to solve.

Input

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

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

The next line contain n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \leq a_i \leq 10^9$, $a_i \neq 0$) separated by spaces describing elements of the array a.

It is guaranteed that the sum of n over all test cases doesn't exceed 10^5.

Output

For each test case, print "YES" if the array can be sorted in the non-decreasing order, otherwise print "NO". You can print each letter in any case (upper or lower).

Example

input
4 7 7 3 2 -11 -13 -17 -23 6 4 10 25 47 71 96 6 71 -35 7 -4 -11 -25 6 -45 9 -48 -67 -55 7
output
NO YES YES NO

Note

In the first test case, there is no way to make the array sorted using the operation any number of times.

In the second test case, the array is already sorted.

In the third test case, we can swap the sign of the 1-st element with the sign of the 5-th element, and the sign of the 3-rd element with the sign of the 6-th element, this way the array will be sorted.

In the fourth test case, there is no way to make the array sorted using the operation any number of times.

B. Dorms War

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

Hosssam decided to sneak into Hemose's room while he is sleeping and change his laptop's password. He already knows the

password, which is a string s of length n. He also knows that there are k *special* letters of the alphabet: $c_1, c_2, \ldots, c_k$.

Hosssam made a program that can do the following.

1. The program considers the current password s of some length m.
2. Then it finds all positions i ($1 \leq i < m$) such that s_{i+1} is one of the k special letters.
3. Then it deletes all of those positions from the password s **even if s_i is a special character**. If there are no positions to delete, then the program displays an error message which has a very loud sound.

For example, suppose the string s is "abcdef" and the special characters are 'b' and 'd'. If he runs the program once, the positions 1 and 3 will be deleted as they come before special characters, so the password becomes "bdef". If he runs the program again, it deletes position 1, and the password becomes "def". If he is wise, he won't run it a third time.

Hosssam wants to know how many times he can run the program on Hemose's laptop without waking him up from the sound of the error message. Can you help him?

Input

The first line contains a single integer t ($1 \leq t \leq 10^5$) — the number of test cases. Then t test cases follow.

The first line of each test case contains a single integer n ($2 \leq n \leq 10^5$) — the initial length of the password.

The next line contains a string s consisting of n lowercase English letters — the initial password.

The next line contains an integer k ($1 \leq k \leq 26$), followed by k distinct lowercase letters $c_1, c_2, \ldots, c_k$ — the special letters.

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

Output

For each test case, print the maximum number of times Hosssam can run the program without displaying the error message, on a new line.

Example

input
10 9 iloveslim 1 s 7 joobeel 2 o e 7 basiozi 2 s i 6 khater 1 r 7 abobeih 6 a b e h i o 5 zondl 5 a b c e f 6 shoman 2 a h 7 shetwey 2 h y 5 samez 1 m 6 mouraz 1 m
output
5 2 3 5 1 0 3 5 2 0

Note

In the first test case, the program can run 5 times as follows: $\text{iloveslim} \to \text{ilovslim} \to \text{iloslim} \to \text{ilslim} \to \text{islim} \to \text{slim}$

In the second test case, the program can run 2 times as follows: $\text{joobeel} \to \text{oel} \to \text{el}$

In the third test case, the program can run 3 times as follows: $\text{basiozi} \to \text{bioi} \to \text{ii} \to \text{i}$.

In the fourth test case, the program can run 5 times as follows: $\text{khater} \to \text{khatr} \to \text{khar} \to \text{khr} \to \text{kr} \to \text{r}$

In the fifth test case, the program can run only once as follows: $\text{abobeih} \rightarrow \text{h}$

In the sixth test case, the program cannot run as none of the characters in the password is a special character.

C. Where is the Pizza?

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

While searching for the pizza, baby Hosssam came across two permutations a and b of length n.

Recall that a permutation is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2,3,1,5,4]$ is a permutation, but $[1,2,2]$ is not a permutation (2 appears twice in the array) and $[1,3,4]$ is also not a permutation ($n=3$ but there is 4 in the array).

Baby Hosssam forgot about the pizza and started playing around with the two permutations. While he was playing with them, some elements of the first permutation got mixed up with some elements of the second permutation, and to his surprise those elements also formed a permutation of size n.

Specifically, he mixed up the permutations to form a new array c in the following way.

- For each i ($1 \le i \le n$), he either made $c_i = a_i$ or $c_i = b_i$.
- The array c is a permutation.

You know permutations a, b, and values at some positions in c. Please count the number different permutations c that are consistent with the described process and the given values. Since the answer can be large, print it modulo 10^9+7.

It is guaranteed that there exists at least one permutation c that satisfies all the requirements.

Input

The first line contains an integer t ($1 \le t \le 10^5$) — the number of test cases.

The first line of each test case contains a single integer n ($1 \le n \le 10^5$) — the length of the permutations.

The next line contains n distinct integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the first permutation.

The next line contains n distinct integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le n$) — the second permutation.

The next line contains n distinct integers $d_1, d_2, \ldots, d_n$ (d_i is either 0, a_i, or b_i) — the description of the known values of c. If $d_i = 0$, then there are no requirements on the value of c_i. Otherwise, it is required that $c_i = d_i$.

It is guaranteed that there exists at least one permutation c that satisfies all the requirements.

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

Output

For each test case, print the number of possible permutations c, modulo 10^9+7.

Example

input
9 7 1 2 3 4 5 6 7 2 3 1 7 6 5 4 2 0 1 0 0 0 0 1 1 1 0 6 1 5 2 4 6 3 6 5 3 1 4 2 6 0 0 0 0 0 8 1 6 4 7 2 3 8 5 3 2 8 1 4 5 6 7 1 0 0 7 0 3 0 5 10 1 8 6 2 4 7 9 3 10 5 1 9 2 3 4 10 8 6 7 5 1 9 2 3 4 10 8 6 7 5 7 1 2 3 4 5 6 7 2 3 1 7 6 5 4 0 0 0 0 0 0 0 5 1 2 3 4 5 1 2 3 4 5 0 0 0 0 0 5 1 2 3 4 5

```
1 2 3 5 4
0 0 0 0 0
3
1 2 3
3 1 2
0 0 0
```

output

```
4
1
2
2
1
8
1
2
2
```

Note

In the first test case, there are 4 distinct permutation that can be made using the process: $[2, 3, 1, 4, 5, 6, 7]$, $[2, 3, 1, 7, 6, 5, 4]$, $[2, 3, 1, 4, 6, 5, 7]$, $[2, 3, 1, 7, 5, 6, 4]$.

In the second test case, there is only one distinct permutation that can be made using the process: $[1]$.

In the third test case, there are 2 distinct permutation that can be made using the process: $[6, 5, 2, 1, 4, 3]$, $[6, 5, 3, 1, 4, 2]$.

In the fourth test case, there are 2 distinct permutation that can be made using the process: $[1, 2, 8, 7, 4, 3, 6, 5]$, $[1, 6, 4, 7, 2, 3, 8, 5]$.

In the fifth test case, there is only one distinct permutation that can be made using the process: $[1, 9, 2, 3, 4, 10, 8, 6, 7, 5]$.

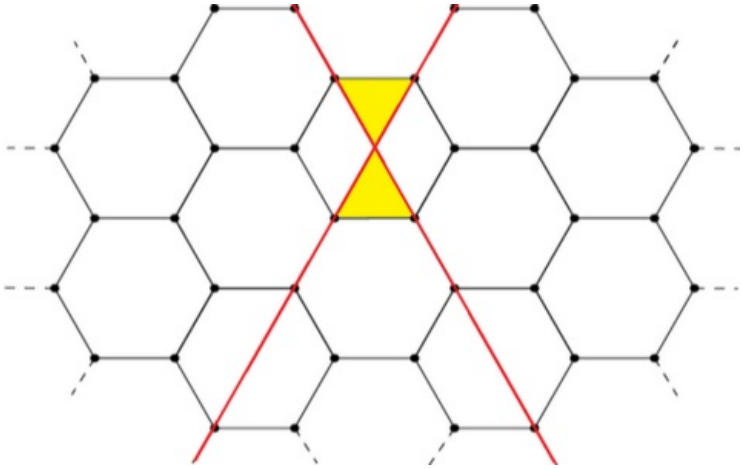
D. Very Suspicious

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

Sehr Sus is an infinite hexagonal grid as pictured below, controlled by MennaFadali, ZerooCool and Hosssam.

They love equilateral triangles and want to create n equilateral triangles on the grid by adding some straight lines. The triangles must all be empty from the inside (in other words, no straight line or hexagon edge should pass through any of the triangles).

You are allowed to add straight lines parallel to the edges of the hexagons. Given n, what is the minimum number of lines you need to add to create at least n equilateral triangles as described?



Adding two red lines results in two new yellow equilateral triangles.

Input

The first line contains a single integer t ($1 \leq t \leq 10^5$) — the number of test cases. Then t test cases follow.

Each test case contains a single integer n ($1 \leq n \leq 10^9$) — the required number of equilateral triangles.

Output

For each test case, print the minimum number of lines needed to have n or more equilateral triangles.

Example

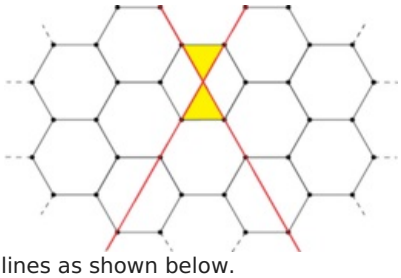
input

```
4
1
2
3
4567
```

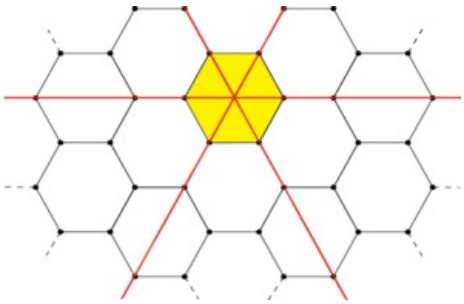
output
2 2 3 83

Note

In the first and second test cases only 2 lines are needed. After adding the first line, no equilateral triangles will be created no matter where it is added. But after adding the second line, two more triangles will be created at once.



In the third test case, the minimum needed is 3 lines as shown below.



E. Hemose on the Tree

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

After the last regional contest, Hemose and his teammates finally qualified to the ICPC World Finals, so for this great achievement and his love of trees, he gave you this problem as the name of his team "Hemose 3al shagra" (Hemose on the tree).

You are given a tree of n vertices where n is a power of 2. You have to give each node and edge an integer value in the range $[1, 2n - 1]$ (inclusive), where all the values are distinct.

After giving each node and edge a value, you should select some root for the tree such that the maximum cost of any simple path starting from the root and ending at any **node or edge** is minimized.

The cost of the path between two nodes u and v or any node u and edge e is defined as the bitwise XOR of all the node's and edge's values between them, including the endpoints (note that in a tree there is only one simple path between two nodes or between a node and an edge).

Input

The first line contains a single integer t ($1 \leq t \leq 5 \cdot 10^4$) — the number of test cases. Then t test cases follow.

The first line of each test case contains a single integer p ($1 \leq p \leq 17$), where n (the number of vertices in the tree) is equal to 2^p.

Each of the next $n - 1$ lines contains two integers u and v ($1 \leq u, v \leq n$) meaning that there is an edge between the vertices u and v in the tree.

It is guaranteed that the given graph is a tree.

It is guaranteed that the sum of n over all test cases doesn't exceed $3 \cdot 10^5$.

Output

For each test case on the first line print the chosen root.

On the second line, print n integers separated by spaces, where the i-th integer represents the chosen value for the i-th node.

On the third line, print $n - 1$ integers separated by spaces, where the i-th integer represents the chosen value for the i-th edge. The edges are numerated in the order of their appearance in the input data.

If there are multiple solutions, you may output any.

Example

input
2 2

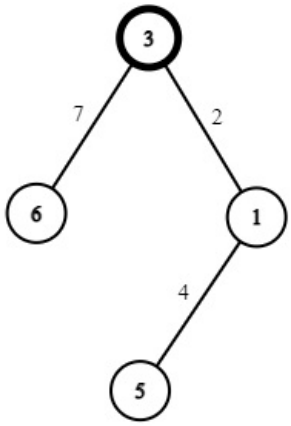
```
1 2
2 3
3 4
3
1 2
2 3
3 4
1 5
1 6
5 7
5 8
```

output

```
3
5 1 3 6
4 2 7
5
1 2 8 11 4 13 9 15
6 14 3 7 10 5 12
```

Note

The tree in the first test case with the weights of all nodes and edges is shown in the picture.

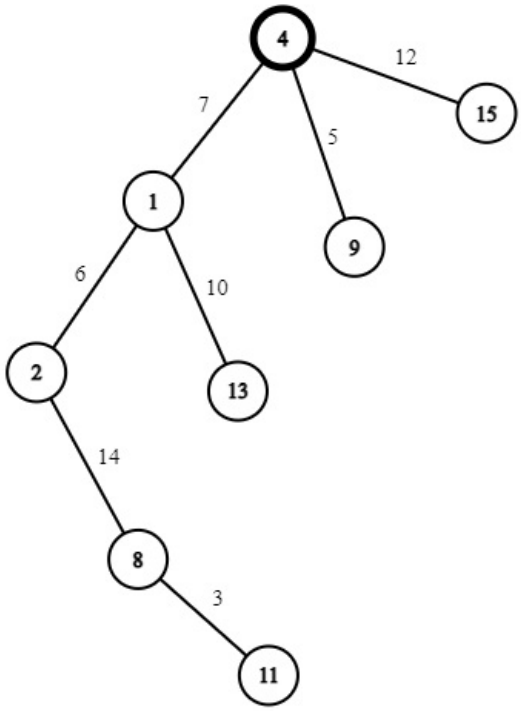


The costs of all paths are:

- 3;
- $3 \oplus 7 = 4$;
- $3 \oplus 7 \oplus 6 = 2$;
- $3 \oplus 2 = 1$;
- $3 \oplus 2 \oplus 1 = 0$;
- $3 \oplus 2 \oplus 1 \oplus 4 = 4$;
- $3 \oplus 2 \oplus 1 \oplus 4 \oplus 5 = 1$.

The maximum cost of all these paths is 4. We can show that it is impossible to assign the values and choose the root differently to achieve a smaller maximum cost of all paths.

The tree in the second test case:



F. Jee, You See?

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

During their training for the ICPC competitions, team "Jee You See" stumbled upon a very basic counting problem. After many "Wrong answer" verdicts, they finally decided to give up and ~~destroy~~ turn-off the PC. Now they want your help in up-solving the problem.

You are given 4 integers n, l, r, and z. Count the number of arrays a of length n containing non-negative integers such that:

- $l \leq a_1 + a_2 + \ldots + a_n \leq r$, and
- $a_1 \oplus a_2 \oplus \ldots \oplus a_n = z$, where $\oplus$ denotes the bitwise XOR operation.

Since the answer can be large, print it modulo $10^9 + 7$.

Input

The only line contains four integers n, l, r, z ($1 \leq n \leq 1000$, $1 \leq l \leq r \leq 10^{18}$, $1 \leq z \leq 10^{18}$).

Output

Print the number of arrays a satisfying all requirements modulo $10^9 + 7$.

Examples

input
3 1 5 1
output
13

input
4 1 3 2
output
4

input
2 1 100000 15629
output
49152

input
100 56 89 66
output
981727503

Note

The following arrays satisfy the conditions for the first sample:

- $[1, 0, 0]$;
- $[0, 1, 0]$;
- $[3, 2, 0]$;
- $[2, 3, 0]$;
- $[0, 0, 1]$;
- $[1, 1, 1]$;
- $[2, 2, 1]$;
- $[3, 0, 2]$;
- $[2, 1, 2]$;
- $[1, 2, 2]$;
- $[0, 3, 2]$;
- $[2, 0, 3]$;
- $[0, 2, 3]$.

The following arrays satisfy the conditions for the second sample:

- $[2, 0, 0, 0]$;
- $[0, 2, 0, 0]$;
- $[0, 0, 2, 0]$;
- $[0, 0, 0, 2]$.