A. Cloudberry Jam

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   512 megabytes
```

The most valuable berry of the Karelian forests is cloudberry. To make jam from cloudberries, you take equal amounts of berries and sugar and cook them. Thus, if you have 2 kg of berries, you need 2 kg of sugar. However, from 2 kg of berries and 2 kg of sugar, you will not get 4 kg of jam, as one might expect, but only 3 kg, since some of the jam evaporates during cooking. Specifically, during standard cooking, exactly a quarter (or 25%) of the jam evaporates.

How many kilograms of cloudberries are needed to prepare n 3-kilogram jars of jam?

Input

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

Each test case contains a single integer n ($1 \le n \le 10^8$) — the number of jars of jam that need to be prepared.

Output

For each test case, output a single integer — the amount of berries needed for the jam in kilograms.

Standard Input	Standard Output
2 1 3	2 6

Note

For the test case 1, explanations are given in the text of the statement — to prepare 1 jar of jam, you need 2 kilograms of cloudberries.

Consider the test case 2: if we take 6 kilograms of berries and 6 kilograms of sugar, we get $\frac{(6+6)\cdot 3}{4} = 9$ kilograms of jam; which gives $\frac{9}{3} = 3$ jars of jam.

,

B. Large Array and Segments

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

There is an array a consisting of n **positive** integers, and a positive integer k. An array b is created from array a according to the following rules:

- b contains $n \cdot k$ numbers;
- the first n numbers of array b are the same as the numbers of array a, that is, $b_i = a_i$ for $i \leq n$;
- for any $i > n$, it holds that $b_i = b_{i-n}$.

For example, if $a = [2, 3, 1, 4]$ and $k = 3$, then $b = [2, 3, 1, 4, 2, 3, 1, 4, 2, 3, 1, 4]$.

Given a number x, it is required to count the number of such positions l ($1 \leq l \leq n \cdot k$), for which there exists a position $r \geq l$, such that the sum of the elements of array b on the segment $[l, r]$ is **at least** x (in other words, $b_l + b_{l+1} + \cdots + b_r \geq x$).

Input

Each test consists of several test cases. The first line contains one integer t ($1 \leq t \leq 10^4$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains three integers n, k, x ($1 \leq n, k \leq 10^5$; $1 \leq x \leq 10^{18}$).

The second line of each test case contains n integers a_i ($1 \leq a_i \leq 10^8$).

Additional constraints on the input:

- the sum of n across all test cases does not exceed $2 \cdot 10^5$;
- the sum of k across all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output one integer — the number of suitable positions l in the array b.

Standard Input	Standard Output
7 5 3 10 3 4 2 1 5 15 97623 1300111 105 95 108 111 118 101 95 118 97 108 111 114 97 110 116 1 100000 1234567891011 1 1 1 1 1 1 1 1 2 2 1 2 1 1	12 1452188 0 1 1 1 0

2 1 5 2 1	

Note

In the first test case, the array b looks like this:

$$[3, 4, 2, 1, 5, 3, 4, 2, 1, 5, 3, 4, 2, 1, 5]$$

There are 12 positions l for which there exists a suitable position r. Here are some (not all) of them:

- $l = 1$, for which there is a position $r = 6$, the sum over the segment $[1, 6]$ equals 18;
- $l = 2$, for which there is a position $r = 5$, the sum over the segment $[2, 5]$ equals 12;
- $l = 6$, for which there is a position $r = 9$, the sum over the segment $[6, 9]$ equals 10.

,

C. Disappearing Permutation

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   512 megabytes
```

A permutation of integers from 1 to n is an array of size n where each integer from 1 to n appears exactly once.

You are given a permutation p of integers from 1 to n. You have to process n queries. During the i-th query, you replace p_{d_i} with 0. Each element is replaced with 0 exactly once. The changes made in the queries are saved, that is, after the i-th query, all integers $p_{d_1}, p_{d_2}, \ldots, p_{d_i}$ are zeroes.

After each query, you have to find the minimum number of operations required to *fix* the array; in other words, to transform the current array into any permutation of integers from 1 to n (possibly into the original permutation p, possibly into some other permutation).

The operation you can perform to fix the array is the following one:

- choose the integer i from 1 to n, replace the i-th element of the array **with** i.

Note that the answer for each query is calculated independently, meaning you do not actually apply any operations, just calculate the minimum number of operations.

Input

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

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

The second line of each test case contains n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$) — the original permutation. All p_i are distinct.

The third line of each test case contains n integers $d_1, d_2, \ldots, d_n$ ($1 \le d_i \le n$). All d_i are distinct.

Additional constraint on the input:

- the sum of n across all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a line containing n integers, where the i-th integer should be equal to the minimum number of operations required to fix the array which was obtained after the i-th query (i.e., the permutation p where all integers $p_{d_1}, p_{d_2}, \ldots, p_{d_i}$ are replaced by zeroes).

Standard Input	Standard Output
3 3 1 2 3 3 2 1 5 4 5 3 1 2	1 2 3 2 4 4 5 5 4 4 4 4 7 7 7

4 5 1 3 2 7 4 3 1 2 7 5 6 1 2 3 4 5 6 7	

Note

In the first test case, after each query, every integer which was replaced by 0 can be restored by one operation.

In the second test case, you can act as follows:

- Query 1: $p = [4, 5, 3, 0, 2]$, it can be transformed into $[1, 5, 3, 4, 2]$.
- Query 2: $p = [4, 5, 3, 0, 0]$, it can be transformed into $[1, 2, 3, 4, 5]$.
- Query 3: $p = [0, 5, 3, 0, 0]$, it can be transformed into $[1, 2, 3, 4, 5]$.
- Query 4: $p = [0, 5, 0, 0, 0]$, it can be transformed into $[1, 2, 3, 4, 5]$.
- Query 5: $p = [0, 0, 0, 0, 0]$, it can be transformed into $[1, 2, 3, 4, 5]$.

The numbers that were changed are highlighted in red.

,

D. Even String

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

You would like to construct a string s, consisting of lowercase Latin letters, such that the following condition holds:

- For every pair of indices i and j such that $s_i = s_j$, the difference of these indices is even, that is, $|i - j| \bmod 2 = 0$.

Constructing any string is too easy, so you will be given an array c of 26 numbers — the required number of occurrences of each individual letter in the string s. So, for every $i \in [1, 26]$, the i-th letter of the Latin alphabet should occur exactly c_i times.

Your task is to count the number of distinct strings s that satisfy all these conditions. Since the answer can be huge, output it modulo $998\,244\,353$.

Input

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

Each test case contains 26 integers c_i ($0 \le c_i \le 5 \cdot 10^5$)— the elements of the array c.

Additional constraints on the input data:

- The sum of c_i for every test case is positive;
- The sum of c_i over all test cases does not exceed $5 \cdot 10^5$.

Output

For each test case, print one integer — the number of suitable strings s, taken modulo $998\,244\,353$.

Standard Input	Standard Output
5 2 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 3 0 0 0 0 0 0 0 0 0 0 0 1 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 233527 233827	4 960 0 1 789493841

Note

In the first test case, there are 4 suitable strings: "abak", "akab", "baka" and "kaba".

,

E. Zebra-like Numbers

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

We call a positive integer *zebra-like* if its binary representation has alternating bits up to the most significant bit, and the least significant bit is equal to 1. For example, the numbers 1, 5, and 21 are *zebra-like*, as their binary representations 1, 101, and 10101 meet the requirements, while the number 10 is not *zebra-like*, as the least significant bit of its binary representation 1010 is 0.

We define the *zebra value* of a positive integer e as the minimum integer p such that e can be expressed as the sum of p *zebra-like* numbers (possibly the same, possibly different)

Given three integers l, r, and k, calculate the number of integers x such that $l \leq x \leq r$ and the *zebra value* of x equals k.

Input

Each test consists of several test cases. The first line contains a single integer t ($1 \leq t \leq 100$) — the number of test cases. The description of test cases follows.

The only line of each test case contains three integers l, r ($1 \leq l \leq r \leq 10^{18}$) and k ($1 \leq k \leq 10^{18}$).

Output

For each test case, output a single integer — the number of integers in $[l, r]$ with *zebra value* k.

Standard Input	Standard Output
5 1 100 3 1 1 1 15 77 2 2 10 100 1234567 123456789101112131 12	13 1 3 0 4246658701

Note

In the first test case, there are 13 suitable numbers: $3, 7, 11, 15, 23, 27, 31, 43, 47, 63, 87, 91, 95$.

Each of them can be represented as a sum of 3 *zebra-like* numbers.

,

F. Online Palindrome

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	512 megabytes

This is an interactive problem.

The jury has a string s consisting of lowercase Latin letters. The following constraints apply to this string:

- the string has an odd length that does not exceed 99;
- the string consists only of the characters "a" and "b".

There is also a string t, which is initially empty. Then, $|s|$ steps happen. During the i-th step, the following events occur:

- first, the jury tells you the character s_i and appends it to the end of the string t;
- then, you may swap any two characters in t, or do nothing.

Your task is to ensure that after the $|s|$-th step, the string t is a palindrome.

Interaction

The interactive interaction starts immediately.

During each step, your program will receive one character — the next character of the string s or "0" (zero) if the string has ended. If the character "0" is received, your program must terminate immediately.

After receiving s_i, you must output the positions of the characters to be swapped, that is, two integers l and r ($1 \le l, r \le i$) or, if you do not want to swap characters, then "0 0".

If your program makes an invalid operation, for example, by trying to swap characters from non-existing positions, or the resulting string t is not a palindrome, the jury program will print one character "X" on a separate line. If your program receives "X" as the response, **it should terminate immediately**; otherwise, the verdict of your submission will be undefined.

The interactor in this problem **is not adaptive**, meaning that the string s does not change during the interaction.

After each output request, do not forget to output a newline and flush the output buffer. Otherwise, you will receive a verdict of "`Idleness limit exceeded`". To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `sys.stdout.flush()` in Python;
- refer to the documentation for other languages.

Hack format

For hacks, use the following format.

The first line should contain a single integer $|s|$ — the length of the string s.

The second line should contain the s itself ($1 \leq |s| \leq 99, |s| \bmod 2 = 1$), consisting of lowercase Latin letters "a" and "b".

Standard Input	Standard Output
a a b 0	0 0 1 2 2 3
a a b a b 0	0 0 2 1 3 2 4 4 4 5