Problem A. Friends of Friends

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second
Memory limit:	64 megabytes

Social networks are very popular now. They use different types of relationships to organize individual users in a network. In this problem friendship is used as a method to connect users. For each user you are given the list of his friends. Consider friendship as a symmetric relation, so if user a is a friend of user b then b is a friend of a.

A friend of a friend for a is such a user c that c is not a friend of a, but there is such b that b is a friend of a and c is a friend of b. Obviously $c \neq a$.

Your task is to find the list of friends of friends for the given user x.

Input

The first line of the input contains integer numbers N and x ($1 \leq N \leq 50$, $1 \leq x \leq N$), where N is the total number of users and x is user to be processed. Users in the input are specified by their numbers, integers between 1 and N inclusive. The following N lines describe friends list of each user. The i-th line contains integer d_i ($0 \leq d_i \leq 50$) — number of friends of the i-th user. After it there are d_i distinct integers between 1 and N — friends of the i-th user. The list doesn't contain i. It is guaranteed that if user a is a friend of user b then b is a friend of a.

Output

You should output the number of friends of friends of x in the first line. Second line should contain friends of friends of x printed in the increasing order.

Examples

stdin	stdout
4 2 1 2 2 1 3 2 4 2 1 3	1 4
4 1 3 4 3 2 3 1 3 4 3 1 2 4 3 1 2 3	0

Problem B: Book Borders

Time limit: 2 s
Memory limit: 512 MiB

A book is being typeset using a fixed width font and a simple greedy algorithm to fill each line. The book contents is just a sequence of words, where each word contains one or more characters.

Before typesetting, we choose a *maximum line length* and denote this value with m. Each line can be at most m characters long including the space characters between the words. The typesetting algorithm simply processes words one by one and prints each word with exactly one space character between two consecutive words on the same line. If printing the word on the current line would exceed the maximum line length m, a new line is started instead.

```
|its.a.long...|          |its.a.long.way|
|way.to.the...|          |to.the.top.if.|
|top.if.you...|          |you.wanna.rock|
|wanna.rock.n.|          |n.roll........|
|roll.........|
```

Text from the example input with maximum line lengths 13 and 14

You are given a text to be typeset and are experimenting with different values of the maximum line length m. For a fixed m, the *leading sentence* is a sentence (a sequence of words separated with a single space character) formed by the first words of lines top to bottom. In the example above, when the sample text is typeset with the maximum line length 14, the leading sentence is "`its to you n`".

Given a text and two integers a and b, find the length of the leading sentence for every candidate maximum line length between a and b inclusive. The length of a sentence is the total number of characters it contains including the space characters.

Input

The first line contains the text to be typeset – a sequence of words separated by exactly one space character. Each word is a string consisting of one or more lowercase letters from the English alphabet.

The second line contains two integers a and b – the edges of the interval we are interested in, as described above.

It is guaranteed that $1 \leq w \leq a \leq b \leq z \leq 500\,000$, where w is the length of the longest word in the text and z is the total number of characters in the text including the space characters.

Output

Output $b - a + 1$ lines – the k-th of those lines should contain a single integer – the total length of the leading sentence when the maximum line length is equal to $a - 1 + k$.

Example

input	output

`its a long way to the top if you wanna rock n roll`
`13 16`
	`22`
	`12`
	`12`
	`15`

Problem C: Kernel Knights

Time limit: 2 s
Memory limit: 512 MiB

Jousting is a medieval contest that involves people on horseback trying to strike each other with wooden lances while riding at high speed. A total of $2n$ knights have entered a jousting tournament – n knights from each of the two great rival houses. Upon arrival, each knight has challenged a single knight from the other house to a duel.

A *kernel* is defined as some subset S of knights with the following two properties:

- No knight in S was challenged by another knight in S.
- Every knight not in S was challenged by some knight in S.

Given the set of the challenges issued, find one kernel. It is guaranteed that a kernel always exists.

Input

The first line contains an integer n ($1 \leq n \leq 100\,000$) – the number of knights of each house. The knights from the first house are denoted with integers 1 through n, knights from the second house with integers $n+1$ through $2n$.

The following line contains integers $f_1, f_2, \ldots, f_n$ – the k-th integer f_k is the index of the knight challenged by knight k ($n+1 \leq f_k \leq 2n$).

The following line contains integers $s_1, s_2, \ldots, s_n$ – the k-th integer s_k is the index of the knight challenged by knight $n+k$ ($1 \leq s_k \leq n$).

Output

Output the indices of the knights in the kernel on a single line. If there is more than one solution, you may output any one.

Example

input

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

output

```
1 2 4 8
```

Problem D: Digit Division

Time limit: 1 s
Memory limit: 512 MiB

We are given a sequence of n decimal digits. The sequence needs to be partitioned into one or more contiguous subsequences such that each subsequence, when interpreted as a decimal number, is divisible by a given integer m.

Find the number of different such partitions modulo $10^9 + 7$. When determining if two partitions are different, we only consider the locations of subsequence boundaries rather than the digits themselves, e.g. partitions 2|22 and 22|2 are considered different.

Input

The first line contains two integers n and m ($1 \leq n \leq 300\,000$, $1 \leq m \leq 1\,000\,000$) – the length of the sequence and the divisor respectively. The second line contains a string consisting of exactly n digits.

Output

Output a single integer – the number of different partitions modulo $10^9 + 7$.

Example

input	input
4 2 1246	4 7 2015
output	**output**
4	0

E. Linear Algebra Test

time limit per test: 3.0 s
memory limit per test: 256 MB
input: standard input
output: standard output

Dr. Wail is preparing for today's test in linear algebra course. The test's subject is `Matrices Multiplication`.

Dr. Wail has n matrices, such that the size of the i^{th} matrix is $(a_i \times b_i)$, where a_i is the number of rows in the i^{th} matrix, and b_i is the number of columns in the i^{th} matrix.

Dr. Wail wants to count how many pairs of indices i and j exist, such that he can multiply the i^{th} matrix with the j^{th} matrix.

Dr. Wail can multiply the i^{th} matrix with the j^{th} matrix, if the number of columns in the i^{th} matrix is equal to the number of rows in the j^{th} matrix.

Input

The first line contains an integer $T\,(1 \leq T \leq 100)$, where T is the number of test cases.

The first line of each test case contains an integer $n\ (1 \leq n \leq 10^5)$, where n is the number of matrices Dr. Wail has.

Then n lines follow, each line contains two integers a_i and $b_i\ (1 \leq a_i, b_i \leq 10^9)\ (a_i \neq b_i)$, where a_i is the number of rows in the i^{th} matrix, and b_i is the number of columns in the i^{th} matrix.

Output

For each test case, print a single line containing how many pairs of indices i and j exist, such that Dr. Wail can multiply the i^{th} matrix with the j^{th} matrix.

Example

input
1 5 2 3 2 3 4 2 3 5 9 4
output
5

Note

As input/output can reach huge size it is recommended to use fast input/output methods: for example, prefer to use `scanf/printf` instead of `cin/cout` in C++, prefer to use `BufferedReader/PrintWriter` instead of `Scanner/System.out` in Java.

In the first test case, Dr. Wail can multiply the 1^{st} matrix (2×3) with the 4^{th} matrix (3×5), the 2^{nd} matrix (2×3) with the 4^{th} matrix (3×5), the 3^{rd} matrix (4×2) with the 1^{st} and second matrices (2×3), and the 5^{th} matrix (9×4) with the 3^{rd} matrix (4×2). So, the answer is 5.

Problem F: Frightful Formula

Time limit: 10 s
Memory limit: 512 MiB

A *frightful matrix* is a square matrix of order n where the first row and the first column are explicitly specified, while the other elements are calculated using a *frightful formula* which is, actually, a simple recursive rule.

Given two integer sequences l and t, both of size n, as well as integer parameters a, b and c, the frightful matrix F is defined as follows:

- The first column of the matrix is the sequence l:

$$F[k,\ 1] = l_k.$$

- The first row of the matrix is the sequence t:

$$F[1,\ k] = t_k.$$

- Other elements are calculated using a recursive formula:

$$F[i,\ j] = a * F[i,\ j-1] + b * F[i-1,\ j] + c.$$

Given a frightful matrix, find the value of the element $F[n, n]$ modulo $10^6 + 3$.

Input

The first line contains four integers n, a, b and c ($2 \le n \le 200\,000,\ 0 \le a,\ b,\ c \le 10^6$) – the size of the matrix and the recursion parameters, as described in the problem statement.

The two following lines contain integers $l_1, \ldots, l_n$ and $t_1, \ldots, t_n$, respectively ($l_1 = t_1,\ 0 \le l_k,\ t_k \le 10^6$).

Output

Output a single integer – the value of $F[n, n]$ modulo $10^6 + 3$.

Example

input

```
3 0 0 0
0 0 2
0 3 0
```

output

```
0
```

input

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

output

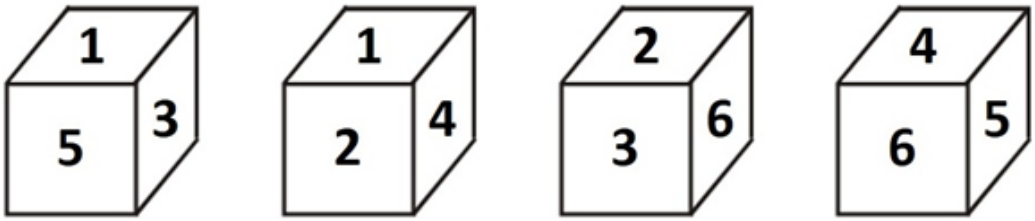
```
41817
```

G. Dice Game

time limit per test: 1.0 s
memory limit per test: 256 MB
input: standard input
output: standard output

A dice is a small cube, with each side having a different number of spots on it, ranging from 1 to 6.

Each side in the dice has 4 adjacent sides that can be reached by rotating the dice (i.e. the current side) 90 degrees. The following picture can help you to conclude the adjacent sides for each side in the dice.



In this problem, you are given a dice with the side containing 1 spot facing upwards, and a sum n, your task is to find the minimum number of required moves to reach the given sum.

On each move, you can rotate the dice 90 degrees to get one of the adjacent sides to the side that currently facing upwards, and add the value of the **new side** to your current sum. According to the previous picture, if the side that currently facing upwards contains 1 spot, then in one move you can move to one of sides that contain 2, 3, 4, or 5 spots.

Initially, your current sum is 0. Even though at the beginning the side that containing 1 spot is facing upwards, but its value will not be added to your sum from the beginning, which means that you must make at least one move to start adding values to your current sum.

Input

The first line contains an integer $T\,(1 \le T \le 200)$, where T is the number of test cases.

Then T lines follow, each line contains an integer $n\ (1 \le n \le 10^4)$, where n is the required sum you need to reach.

Output

For each test case, print a single line containing the minimum number of required moves to reach the given sum. If there is no answer, print -1.

Example

input
2 5 10
output
1 2

Note

In the first test case, you can rotate the dice 90 degrees one time, and make the side that contains 5 spots facing upwards, which make the current sum equal to 5. So, you need one move to reach sum equal to 5.

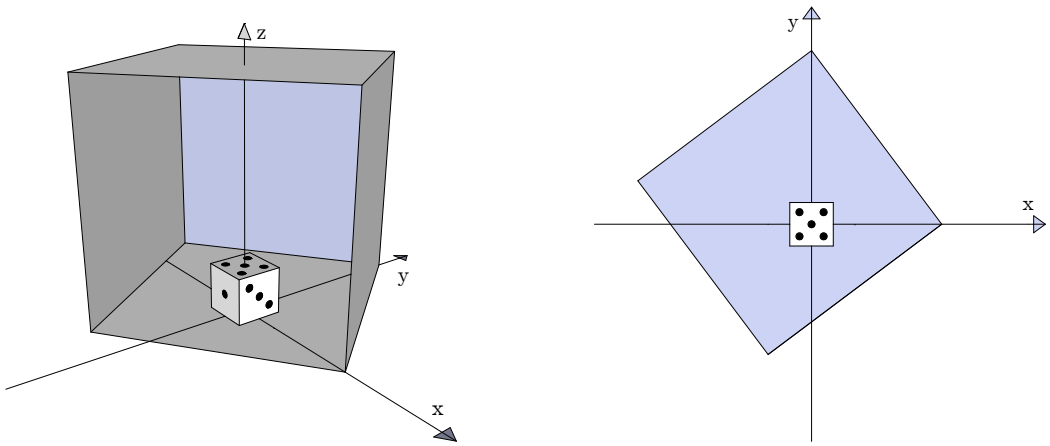
In the second test case, you can rotate the dice 90 degrees one time, and make the side that contains 4 spots facing upwards, which make the current sum equal to 4. Then rotate the dice another 90 degrees, and make the side that contains 6 spots facing upwards, which make the current sum equal to 10. So, you need two moves to reach sum equal to 10.

Problem H: Hovering Hornet

Time limit: 1 s
Memory limit: 512 MiB

You have managed to trap a hornet inside a box lying on the top of your dining table. Unfortunately, your playing dice is also trapped inside – you cannot retrieve it and continue your game of Monopoly without risking the hornet's wrath. Instead, you pass your time calculating the expected number of spots on the dice visible to the hornet.

The hornet, the dice and the box are located in the standard three-dimensional coordinate system with the x coordinate growing eastwards, the y coordinate growing northwards and the z coordinate growing upwards. The surface of the table corresponds to the x-y plane.



Perspective and the birds-eye view of the second example input

The dice is a $1 \times 1 \times 1$ cube, resting on the table with the center of the bottom side exactly in the origin. Hence, the coordinates of its two opposite corners are $(-0.5,\ -0.5,\ 0)$ and $(0.5,\ 0.5,\ 1)$. The top side of the dice has 5 spots, the south side 1 spot, the east side 3 spots, the north side 6 spots, the west side 4 spots and the (invisible and irrelevant) bottom side 2 spots.

The box is a $5 \times 5 \times 5$ cube also resting on the table with the dice in its interior. The box is specified by giving the coordinates of its bottom side – a 5×5 square.

Assume the hornet is hovering at a uniformly random point in the (continuous) space inside the box not occupied by the dice. Calculate the expected number of spots visible by the hornet. The dice is opaque and, hence, the hornet sees a spot only if the segment connecting the center of the spot and the location of the hornet does not intersect the interior of the dice.

Input

Input consists of 4 lines. The k-th line contains two floating-point numbers x_k and y_k ($-5 \leq x_k,\ y_k \leq 5$) – coordinates of the k-th corner of the bottom side of the box in the x-y plane. The coordinates are given in the counterclockwise direction and they describe a square with the side length of exactly 5.

The box fully contains the dice. The surfaces of the box and the dice do not intersect or touch except along the bottom sides.

Output

Output a single floating point number – the expected number of spots visible. The solution will be accepted if the absolute or the relative difference from the judges solution is less than 10^{-6}.

Example

input

```
-2.5 -1.5
2.5 -1.5
2.5 3.5
-2.5 3.5
```

output

```
10.6854838710
```

input

```
3 0
0 4
-4 1
-1 -3
```

output

```
10.1226478495
```