Problem A. Analyzing Bit (Yet Special) Strings

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

Do you think that analyzing bit strings is easy? This is not the case when you are in a dream.

So you are in a dream. Unexpectedly, right? But... I am afraid it's not the dream you always wanted to be in. You have a string of bits in your dream, a long string of bits you are to deal with. And you clearly understand what you should do to leave this horrible dream right now: find the best *special* string.

Luckily, you know that you read a book about the theory of *special* strings yesterday. You only managed to remember the strangest definition of *special* strings, though, which sounded as follows.

Suppose you have a string of bits T of length n. Bits of T are referred to as T_1, T_2, ..., T_n. Let's denote $A(i, j)$ and $B(i, j)$ as the number of $0-$bits and $1-$bits among T_i, T_{i+1}, ..., T_j, correspondingly. String T is called *special* if for every i between 1 and n, inclusive, both of the following conditions hold: $A(1, i) \geq B(1, i)$ and $A(i, n) \leq B(i, n)$.

But you can't be satisfied with just any *special* string. You need the best *special* string. The dream was very strange and thus the rules to determine which of two strings was better were strange as well. Let L_1 and L_2 be the lengths of two strings, and P_1 and P_2 be their numbers of occurrences in the given string S as a substring, respectively. Then you know that first string is better than the second one if $L_1 \cdot P_1 > L_2 \cdot P_2$.

So your task is simple... or not? Find the best *special* string — a *special* string such that no other *special* string is better.

Input

The only line of the input file contains S ($2 \leq |S| \leq 2 \cdot 10^5$) — the string consisting of zeroes and ones.

Output

The first line of the output file should contain the value of $L \cdot P$, where L is the length of the best *special* string and P is the number of its occurrences in S as a substring. The second line of the output file should contain the best *special* string itself. If there are several best *special* strings, you can choose any of them.

It is guaranteed that at least one *special* string is a substring of S.

Examples

standard input	standard output
00111001110101	8 0011
00011001110101	14 00011001110101
0101010101	18 010101

Problem B. Bits Are Dangerous

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

Do you think that escaping a dream is easy? This is not the case when the dream is about bits.

You've surely read too much about bit theory yesterday, but this doesn't matter now as the only thing you want to do is to escape from the dream. You've probably found the best *special* string incorrectly... or they've just cheated you. You still have a long string of bits in front of your closed eyes.

Then you realize that you have an ability to change the leftmost bit of this string from 0 to 1 or from 1 to 0. It takes you exactly four seconds, but you can!

Then you realize that you can also shift this string cyclically, one position to the left or one position to the right. Either of the actions takes you a whopping total of seven seconds in this strange dream.

Something tells you that those 1-bits in the string are what's keeping you in the dream. As this might be true, you decide to turn the string into a string of all zeroes — you know that you have everything required for that.

But how much time would it take you if you acted optimally?

Input

The only line of the input file contains S ($2 \le |S| \le 2 \cdot 10^5$) — the string consisting of zeroes and ones.

Output

Output the sought minimum time you need to turn S into a string of all zeroes, in seconds.

Examples

standard input	standard output
11001	33
01101010	58
1000110101	62
011010100011	94
0000	0

Note

In the first test case, the optimal strategy is the following: change the leftmost bit to get 01001, shift the string cyclically to the left to get 10010, change the leftmost bit to get 00010, shift the string cyclically to the right to get 00001, shift the string cyclically to the right (again) to get 10000, and then change the leftmost bit to get 00000. Three changes and three cyclic shifts would take you exactly 33 seconds.

Problem C. Counting Amusing Numbers

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

Do you think that counting is easy? This is not the case when you don't fully understand objects that you are counting.

Let's call a $2N$-digit integer X (possibly with leading zeroes) *amusing* if two N-digit integers a and b (again, possibly with leading zeroes) exist such that $a + b = 10^N$ and $S_d(X) = S_d(a) + S_d(b)$ holds for every digit d, where $S_d(P)$ $(0 \le d \le 9)$ is the number of occurrences of digit d in the decimal representation of P. For example, numbers 46 $(4 + 6 = 10^1)$, 9820 $(98 + 02 = 10^2)$ and 08362090 $(6020 + 3980 = 10^4)$ are amusing.

You are given a sequence of digits and question marks of an even length. Find the number of ways to replace question marks with digits to get an amusing number, modulo $10^9 + 7$.

Input

The only line of the input file contains a non-empty sequence of digits and question marks. The length of the sequence is even and doesn't exceed 10^5. The number of question marks doesn't exceed 1000.

Output

Output the sought number of ways modulo $10^9 + 7$.

Examples

standard input	standard output
2?4?	4
29?2?3	0
?820	2

Problem D. Degree Of Number's Eccentricity

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

Do you think that being eccentric is easy? This is not the case when you're a number.

The *degree of eccentricity* of a $2N$-digit integer X (possibly with leading zeroes) is defined as the smallest possible value of $|a + b - 10^N|$ for some N-digit integers a and b (again, possibly with leading zeroes) such that $S_d(X) = S_d(a) + S_d(b)$ holds for every digit d, where $S_d(P)$ $(0 \le d \le 9)$ is the number of occurrences of digit d in the decimal representation of P. For example, the degree of eccentricity of *amusing* numbers (see problem *Counting Amusing Numbers*) is equal to 0, while the degree of eccentricity of 192747 is equal to 7 ($|274 + 719 - 1000| = 7$).

You are given a bunch of numbers of even lengths. Find the degree of eccentricity of each of them.

Input

The first line of the input file contains the number of test cases T $(1 \le T \le 1000)$. Each of the next T lines contains an integer number of an even length (possibly with leading zeroes). The total length of all numbers in the input file (except T) doesn't exceed 10^6.

Output

For each test case, output one line containing the degree of eccentricity of the corresponding number in the input file.

Examples

standard input	standard output
3 9820 192747 000001	0 7 900

Problem E. Exact Number of Drops

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

Do you think that managing a restaurant is easy? That is not the case when all clients are pretty demanding.

You are managing a restaurant, and every day a lot of people come and ask you to give them some drinks. As your restaurant has just recently opened, you don't have any vessels except two, which can accommodate a drops of liquid and b drops of liquid, respectively.

In order not to make your clients wait (or at least to minimize their waiting time), you always have to determine the minimal number of steps required to obtain exactly c drops of liquid in one of the vessels.

At the beginning both vessels are empty. The following operations are counted as *steps*:

- emptying a vessel;
- filling a vessel;
- pouring liquid from one vessel to the other (without spilling) until one of the vessels is either full or empty.

Eventually you became tired to do it all by hand and decided to write a program which will help you calculate the minimum number of steps required in all these cases.

Input

The first line of the input file contains the number of test cases T ($1 \le T \le 10^5$). The next T lines contain one testcase each and consist of three integers a, b and c ($1 \le a, b, c \le 10^9$).

Output

For each test case, output the minimum number of steps required to obtain exactly c drops of liquid in one of the vessels or -1 if this is impossible.

Examples

standard input	standard output
2 2 5 4 5 3 7	4 -1

Problem F. Figures Of Simple Sense

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

Do you think that drawing polygons is easy? This is not the case when you have some restrictions.

In this problem all you have to do is just to draw a polygon. It must have exactly N vertices. It must contain no self-intersections. No three consecutive vertices of the polygon must be collinear. All coordinates of its vertices must be integers between 0 and 10 000, inclusive. Easy, right?

There is one more small restriction though. The number of inner angles of this polygon equal to 90° must be as large as possible under these constraints. What do you think about it now?

Input

The input file contains the number of test cases T ($1 \le T \le 30$) followed by T integer numbers N ($3 \le N \le 1000$).

Output

For each test case, output the maximum possible number of inner angles equal to 90° followed by N pairs of integers — the coordinates of the vertices of the polygon in either clockwise or counterclockwise order. Of course, more than one solution is possible, so output any of them.

Examples

standard input	standard output
2 4 6	4 0 0 1 0 1 1 0 1 5 1 1 3 3 5 1 4 0 3 1 2 0

Problem G. Gowk's Errand for Master

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

Do you think that sorting is easy? This is not the case when you're running out of time.

Unexpected weird things often happen in our life. For example, now you have array A and you need to sort it in non-decreasing order. The problem is that you don't really have time to do that.

Fortunately, your friend is very good at sorting arrays, so you decided to ask for his help. You believe that your friend's special abilities are due to his very conceptual approach. In one second he takes an element out of the array and places it either at the beginning or the end of the array. For example, if the array is 4 2 5 6 1 3, then taking out 5 and placing it at the beginning would yield 5 4 2 6 1 3, while taking out 2 and placing it at the end would yield 4 5 6 1 3 2.

Of course, your friend does everything as fast as possible, because he doesn't want to waste his time. The only thing you want to know is how much time you have for doing other useful things while your array is being sorted.

Input

The first line of the input file contains N — the number of elements in A ($1 \le N \le 3 \cdot 10^5$). The second line contains N integer numbers A_i ($1 \le A_i \le 10^6$).

Output

Output the sought minimum time needed for your friend to sort A, in seconds.

Examples

standard input	standard output
5 2 5 1 3 3	2
4 1 2 2 1	1

Note

Your friend will sort the array from the first example test case in two seconds: he'll first move 1 to the beginning to yield 1 2 5 3 3, and then he'll move 5 to the end to yield 1 2 3 3 5.

Problem H. Handicapped Onsite Prediction

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

Do you think that winning a competition is easy? This is not the case when there are so many incredible competitors around.

You're taking part in the OpenBowl programming contest. It consists of two rounds — online round and onsite round. Besides you, there are $N - 1$ more competitors eager to win. Each of N competitors has already taken part in the online round, and competitor i received exactly A_i points (you have no idea about how these numbers were calculated — only Sn., the main contest organizer, knows everything about that; you've only heard that it had something to do with *conditionally unrated rounds*).

Now it's time for the onsite round. At the onsite round every competitor takes some place between 1 and N, inclusive, and no two contestants take the same place. For place j at the onsite round P_j points are awarded, and the final score of the contestant is equal to the sum of points received by him during the online and the onsite rounds. Then each competitor's final place is calculated — for competitor i it is equal to $k + 1$, where k is the number of competitors whose final score is strictly greater than that of competitor i.

You clearly understand that your rivals are very strong. That's why you aren't even aiming at winning the contest. You decided that you will be pleased with your result if the final place you take is not lower than X. Now you would like to find out: what is the lowest place at the onsite round you should take to guarantee that?

Input

The input file contains two integer numbers N and X ($1 \le X \le N \le 10^5$), followed by N integer numbers A_i — the number of points competitor i received during the online round, followed by N integer numbers P_j — the number of points received by the competitor who takes place j at the onsite round ($0 \le A_i, P_j \le 10^9$). It is guaranteed that $P_j \ge P_{j+1}$ for any j, $1 \le j < N$. You are competitor number 1.

Output

Output one integer number between 1 and N, inclusive — the lowest place at the onsite round you should take in order to guarantee taking place X or higher overall, or -1 if it is impossible to have such a guarantee even in case of winning the onsite round.

Examples

standard input	standard output
5 3 230 310 200 260 180 100 80 60 50 45	2
5 2 230 310 200 260 180 100 80 60 50 45	-1

Problem I. Icy Roads Of Nomel

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

Do you think that driving is easy? This is not the case when the roads are covered with ice.

The city of Nomel has exactly $N + 1$ streets (west-east roads) and $M + 1$ avenues (north-south roads). Each street intersects each avenue, so each street is divided into M blocks and each avenue is divided into N blocks.

It's winter, and each road of the city is covered with a thick ice layer. As driving on ice is a bit tricky, each road has its own time of driving through one block of this road. This value is constant along all blocks of the road.

Obviously, you can drive only through the roads. Now your task is to find a route from the north-western intersection of the city to the south-eastern one with the minimum driving time. This route must also be the shortest one, that is, it should pass through exactly $N + M$ blocks.

Input

The first line of the input file contains two integers N and M ($1 \le N, M \le 500\,000$). The second line contains $N + 1$ positive integers representing the driving times of Nomel streets, given in the order from north to south. The third line contains $M + 1$ positive integers representing the driving times of Nomel avenues, given in the order from west to east. It's guaranteed that none of these numbers exceed 10^9.

Output

Output one integer — the required minimum total driving time.

Examples

standard input	standard output
2 3 5 3 7 7 2 5 6	19

Problem J. Jelly-Oxygen Beans

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

Do you think that eating candies is easy? This is not the case when they are oxygen candies actually.

As you like everything sweet, you've just bought a fresh pack of N jelly beans. But usual jelly beans are, of course, not an option. As a part of your desire to taste everything in your life, you've bought special jelly-oxygen beans, rare and exclusive candies.

Now it came to eating, and you decided to solve the jelly-oxygen beans eating problem in a mathematical way.

Suppose you want to eat N jelly-oxygen beans during the next M ($1 \le M \le N$) days, eating the same number of jelly-oxygen beans each day. It might be impossible, however, if M doesn't divide N. In this case, you want to eat $\lfloor N/M \rfloor$ jelly-oxygen beans each day. The remaining $N \bmod M$ jelly-oxygen beans should be divided into M equal smaller parts. If this is possible, you'll eat exactly one of these parts each day.

How many possible choices of M do you have?

Input

The only line of the input file contains an integer number N ($1 \le N \le 10^{12}$).

Output

Output the number of possible values of M.

Examples

standard input	standard output
5	4

Note

The possible values of M in the example are 1 (eat all candies on the only day), 2 (divide a candy into two equal parts and eat two undivided candies and one of the parts each day), 4 (divide a candy into four equal parts and eat one undivided candy and one of the parts each day) and 5 (eat one candy each day). Note that M can't be equal to 3, as you can't divide $N \bmod M = 2$ candies into $M = 3$ equal parts.