A.CONCATENATION

Time limit: 2s

Memory limit: 256 MB

Famous programmer Gennady Korotkevich likes to create new words. One way to do it is to concatenate existing words. That means writing one word after another. For example, if he has words “cat” and “dog”, he would get a word “catdog”, that could mean something like the name of a creature with two heads: one cat head and one dog head.

Gennady is a bit bored of this way of creating new words, so he has invented another method. He takes a non-empty prefix of the first word, a non-empty suffix of the second word, and concatenates them. For example, if he has words “tree” and “heap”, he can get such words as “treap”, “tap”, or “theap”. Who knows what they could mean?

Gennady chooses two words and wants to know how many different words he can create using his new method. Of course, being a famous programmer, he has already calculated the answer. Can you do the same?

Input

Two lines of the input file contain words chosen by Gennady. They have lengths between 1 and 100 000 characters and consist of lowercase English letters only.

Output

Output one integer — the number of different words Gennady can create out of words given in the input file.

Example

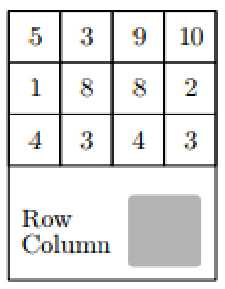
Sample Input	**Sample Output**
cat dog	9
tree heap	14

B.LUCKY CHANCES

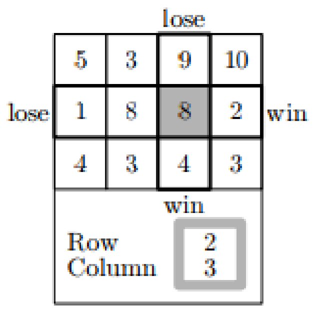
Time limit: 2s

Memory limit: 256 MB

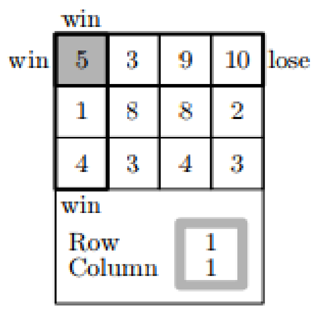
Lucky Chances is a lottery game. Each lottery ticket has a play field and a scratch area. The play field is a rectangular $r \times c$ field filled with numbers. The scratch area hides row and column numbers that specify the bet cell.



Unscratched ticket



Scratched ticket 1



Scratched ticket 2

There are four possible winning directions: up, down, left and right. You win a direction if all numbers in this direction from the bet cell are strictly less than a number in the bet cell. And if the bet cell is on the edge of the grid, you win the corresponding direction automatically!

Larry wants to choose the ticket that has maximum total number of winning directions for all possible bet cells. Write a program that determines this number for the given grid.

Input

The first line of the input file contains two integers r and c — the number of rows and columns in the grid ($1 \le r, c \le 100$). The following r lines contain c integers each — the numbers printed on the grid. Each number is positive and does not exceed 1000.

Output

Output a single integer w — the total number of winning directions for the given grid.

Example

Sample Input	Sample Output
3 4 5 3 9 10	25

1 8 8 2 4 3 4 3	

C.JOURNEY TO THE “THE WORLD’S START”

Time limit: 2s

Memory limit: 256 MB

Jerry Prince is the fourth grade student and he goes to New-Lodnon to visit the most popular amusement park “The World’s Start”.

An airport he arrives at is next to the first stop of the metro line. This line has n stops and “The World’s Start” is on the last of them. The metro of New-Lodnon is pretty fast so you may assume that you can get from a stop to the next one in just one minute.

Jerry needs a travel card to use the metro. Each travel card has a range r and a price p. With a travel card of range r Jerry may travel no more than r stops at once. Therefore, if Jerry enters metro at the stop i he should exit on one of the stops from $i - r$ to $i + r$ inclusive. It takes d_i minutes to exit and reenter metro at *i-th* stop. There is no time required to enter the first stop or exit the last one.

Jerry is not very rich but he has some spare time, so he decided to buy the cheapest travel card that will allow him to travel from the first metro stop to the last one in no more than t minutes.

Input

The first line of the input file contains two integers n and t — the number of stops and the maximum possible time ($2 \leq n \leq 50000$; $n - 1 \leq t \leq 10^9$).

The second line contains $n - 1$ integers p_r — the prices of travel cards with range $r = 1 \ldots n - 1$ ($1 \leq p_r \leq 100000$).

The third line contains $n - 2$ integers d_i — the number of minutes required to re-enter metro at stop $i = 2 \ldots n - 1$ ($1 \leq d_i \leq 100000$).

Output

Output a single integer p — the lowest possible price of one travel card that allows Jerry to travel from the first to the last stop in no more than t minutes.

Example

Sample Input	**Sample Output**
4 4	2
1 2 3	
1 4	

D.3-DIVISIBLE PAIRS

Time limit: 1s

Memory limit: 128 MB

You are given an array of N positive integers. Compute the number of pairs of integers in the array that have the sum divisible by 3.

Input

The first line contains one integer N ($1 \leq N \leq 100000$).

The second line contains N integers representing the elements of the array ($1 \leq element \leq 100000$).

Output

Output a single number representing the number of pairs that have the sum divisible by 3.

Examples

Sample Input	**Sample Output**
5 1 4 2 3 3	3
4 3 3 3 6	6
8 1 1 7 2 11 3 6 9	9

E.KINGDOM TRIP

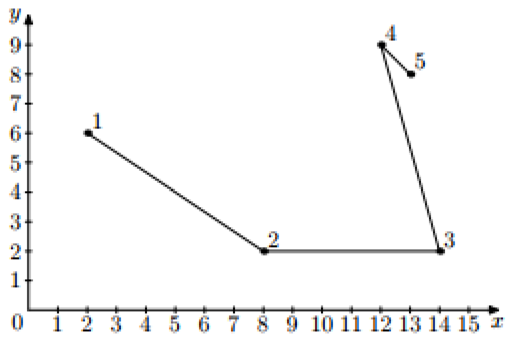
Time limit: 1s

Memory limit: 128 MB

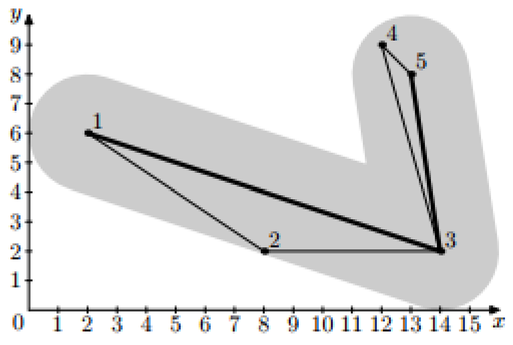
Once upon a time, there was a kingdom ruled by a wise king. After forty three years of his reign, by means of successful military actions and skillful diplomacy, the kingdom became an infinite flat two-dimensional surface. This form of the kingdom greatly simplified travelling, as there were no borders.

A big holiday was planned in the kingdom. There were n locations for people to gather. As the king wanted to have a closer look at his people, he ordered to make a trip through these locations. He wanted to give a speech in each of these locations. Initially his trip was designed as a polygonal chain p: $p_1 \to p_2 \to . . . \to p_n$.

Not only the king was wise, but he was old, too. Therefore, his assistants came up with an idea to skip some locations, to make the king to give as few speeches as possible. The new plan of the trip has to be a polygonal chain consisting of some subsequence of p: starting at p_1 and ending at p_n, formally, $p_{i1} \to p_{i2} \to \cdot \cdot \cdot \to p_{im}$, where $1 = i_1 < i_2 < ... < i_m = n$. Assistants know that the king wouldn't allow to skip location j, if the distance from p_j to segment $p_{ik} \to p_{ik+1}$ exceeds d, for such k, that $i_k < j < i_{k+1}$.



Original route



New route

Help the assistants to find the new route that contains the minimum possible number of locations.

Input

The first line of the input file contains two integers n and d — the number of locations in the initial plan of the trip and the maximum allowed distance to skipped locations ($2 \le n \le 2000$; $1 \le d \le 10^6$). The following n lines describe the trip. The *i-th* of these lines contains two integers x_i and y_i — coordinates of point p_i . The absolute value of coordinates does not exceed 10^6. No two points coincide.

Output

Output the minimum number of locations the king will visit. It is guaranteed that the answer is the same for $d \pm 10^{-4}$.

Example

Sample Input	Sample Output

5 2
2 6
8 2
14 2
12 9
13 8
	3

F.Lineland Mail

Time limit: 3s

Memory limit: 128 MB

All cities of Lineland are located on the *Ox* coordinate axis. Thus, each city is associated with its position x_i — a coordinate on the *Ox* axis. No two cities are located at a single point.

Lineland residents love to send letters to each other. A person may send a letter only if the recipient lives in another city (because if they live in the same city, then it is easier to drop in). Strange but true, the cost of sending the letter is exactly equal to the distance between the sender's city and the recipient's city.

For each city calculate two values min_i and max_i, where min_i is the minimum cost of sending a letter from the *i*-th city to some other city, and max_i is the the maximum cost of sending a letter from the *i*-th city to some other city.

Input

The first line of the input contains integer n ($2 \le n \le 10^5$) — the number of cities in Lineland. The second line contains the sequence of n distinct integers $x_1, x_2, ..., x_n$ ($-10^9 \le x_i \le 10^9$), where x_i is the *x*-coordinate of the *i*-th city. All the x_i's are distinct and follow in **ascending** order.

Output

Print n lines, the *i*-th line must contain two integers min_i, max_i, separated by a space, where min_i is the minimum cost of sending a letter from the *i*-th city, and max_i is the maximum cost of sending a letter from the *i*-th city.

Example

Sample Input	Sample Output

4
-5 -2 2 7
	3 12
	3 9
	4 7
	5 12

Sample Input	**Sample Output**
2 -1 1	2 2 2 2

G. Sasha and Sticks

Time limit: 2s

Memory limit: 256 MB

It's one more school day now. Sasha doesn't like classes and is always bored at them. So, each day he invents some game and plays in it alone or with friends.

Today he invented one simple game to play with Lena, with whom he shares a desk. The rules are simple. Sasha draws n sticks in a row. After that the players take turns crossing out exactly k sticks from left or right in each turn. Sasha moves first, because he is the inventor of the game. If there are less than k sticks on the paper before some turn, the game ends. Sasha wins if he makes strictly more moves than Lena. Sasha wants to know the result of the game before playing, you are to help him.

Input

The first line contains two integers n and k ($1 \le n, k \le 10^{18}$, $k \le n$) — the number of sticks drawn by Sasha and the number k — the number of sticks to be crossed out on each turn.

Output

If Sasha wins, print "YES" (without quotes), otherwise print "NO" (without quotes).

You can print each letter in arbitrary case (upper of lower).

Example

Sample Input	**Sample Output**
1 1	YES

Sample Input	**Sample Output**
10 4	NO

H.Permutation

Time limit: 2s

Memory limit: 256 MB

"Hey, it's homework time" — thought Polycarpus and of course he started with his favourite subject, IT. Polycarpus managed to solve all tasks but for the last one in 20 minutes. However, as he failed to solve the last task after some considerable time, the boy asked you to help him.

The sequence of *n* integers is called a permutation if it contains all integers from 1 to *n*exactly once.

You are given an arbitrary sequence $a_1, a_2, ..., a_n$ containing *n* integers. Each integer is not less than 1 and not greater than 5000. Determine what minimum number of elements Polycarpus needs to change to get a permutation (he should not delete or add numbers). In a single change he can modify any single sequence element (i. e. replace it with another integer).

Input

The first line of the input data contains an integer *n* ($1 \le n \le 5000$) which represents how many numbers are in the sequence. The second line contains a sequence of integers a_i($1 \le a_i \le 5000$, $1 \le i \le n$).

Output

Print the only number — the minimum number of changes needed to get the permutation.

Example

Sample Input	**Sample Output**
3 3 1 2	0

Sample Input	**Sample Output**
2	1

2 2	

Sample Input	Sample Output
5 5 3 3 3 1	2