CODEFORCES$^{\beta}$
Sponsored by Telegram

Codeforces Round #153 (Div. 2)

A. Little Xor

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

Little Petya likes arrays that consist of non-negative integers a lot. Recently his mom has presented him one such array consisting of n elements. Petya immediately decided to find there a segment of consecutive elements, such that the xor of all numbers from this segment was maximal possible. Help him with that.

The xor operation is the bitwise exclusive "`OR`", that is denoted as "`xor`" in Pascal and "`^`" in C/C++/Java.

Input

The first line contains integer n $(1 \le n \le 100)$ — the number of elements in the array. The second line contains the space-separated integers from the array. All numbers are non-negative integers strictly less than 2^{30}.

Output

Print a single integer — the required maximal xor of a segment of consecutive elements.

Sample test(s)

input

```
5
1 2 1 1 2
```

output

```
3
```

input

```
3
1 2 7
```

output

```
7
```

input

```
4
4 2 4 8
```

output

```
14
```

Note

In the first sample one of the optimal segments is the segment that consists of the first and the second array elements, if we consider the array elements indexed starting from one.

The second sample contains only one optimal segment, which contains exactly one array element (element with index three).

B. Unsorting Array

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

Little Petya likes arrays of integers a lot. Recently his mother has presented him one such array consisting of n elements. Petya is now wondering whether he can swap any two distinct integers in the array so that the array got unsorted. Please note that Petya can not swap equal integers even if they are in distinct positions in the array. Also note that Petya **must** swap some two integers even if the original array meets all requirements.

Array a (the array elements are indexed from 1) consisting of n elements is called sorted if it meets at least one of the following two conditions:

1. $a_1 \le a_2 \le ... \le a_n$;
2. $a_1 \ge a_2 \ge ... \ge a_n$.

Help Petya find the two required positions to swap or else say that they do not exist.

Input

The first line contains a single integer n $(1 \le n \le 10^5)$. The second line contains n non-negative space-separated integers $a_1, a_2, ..., a_n$ — the elements of the array that Petya's mother presented him. All integers in the input do not exceed 10^9.

Output

If there is a pair of positions that make the array unsorted if swapped, then print the numbers of these positions separated by a space. If there are several pairs of positions, print any of them. If such pair does not exist, print -1. The positions in the array are numbered with integers from 1 to n.

Sample test(s)

input

```
1
1
```

output

```
-1
```

input

```
2
1 2
```

output

```
-1
```

input

```
4
1 2 3 4
```

output

```
1 2
```

input

```
3
1 1 1
```

output

```
-1
```

Note

In the first two samples the required pairs obviously don't exist.

In the third sample you can swap the first two elements. After that the array will look like this: `2 1 3 4`. This array is unsorted.

C. Points on Line

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

Little Petya likes points a lot. Recently his mom has presented him n points lying on the line OX. Now Petya is wondering in how many ways he can choose three distinct points so that the distance between the two farthest of them doesn't exceed d.

Note that the order of the points inside the group of three chosen points doesn't matter.

Input

The first line contains two integers: n and d ($1 \le n \le 10^5$; $1 \le d \le 10^9$). The next line contains n integers $x_1, x_2, ..., x_n$, their absolute value doesn't exceed 10^9 — the x-coordinates of the points that Petya has got.

It is guaranteed that the coordinates of the points in the input **strictly increase**.

Output

Print a single integer — the number of groups of three points, where the distance between two farthest points doesn't exceed d.

Please do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Sample test(s)

input
```
4 3
1 2 3 4
```
output
```
4
```

input
```
4 2
-3 -2 -1 0
```
output
```
2
```

input
```
5 19
1 10 20 30 50
```
output
```
1
```

Note

In the first sample any group of three points meets our conditions.

In the seconds sample only 2 groups of three points meet our conditions: `{-3, -2, -1}` and `{-2, -1, 0}`.

In the third sample only one group does: `{1, 10, 20}`.

D. Playing with Permutations

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

Little Petya likes permutations a lot. Recently his mom has presented him permutation $q_1, q_2, ..., q_n$ of length n.

A *permutation* a of length n is a sequence of integers $a_1, a_2, ..., a_n$ $(1 \leq a_i \leq n)$, all integers there are distinct.

There is only one thing Petya likes more than permutations: playing with little Masha. As it turns out, Masha also has a permutation of length n. Petya decided to get the same permutation, whatever the cost may be. For that, he devised a game with the following rules:

- Before the beginning of the game Petya writes permutation $1, 2, ..., n$ on the blackboard. After that Petya makes exactly k moves, which are described below.
- During a move Petya tosses a coin. If the coin shows heads, he performs point 1, if the coin shows tails, he performs point 2.

1. Let's assume that the board contains permutation $p_1, p_2, ..., p_n$ at the given moment. Then Petya removes the written permutation p from the board and writes another one instead: $p_{q_1}, p_{q_2}, ..., p_{q_n}$. In other words, Petya applies permutation q (which he has got from his mother) to permutation p.
2. All actions are similar to point 1, except that Petya writes permutation t on the board, such that: $t_{q_i} = p_i$ for all i from 1 to n. In other words, Petya applies a permutation that is inverse to q to permutation p.

We know that after the k-th move the board contained Masha's permutation $s_1, s_2, ..., s_n$. Besides, we know that throughout the game process Masha's permutation **never occurred on the board** before the k-th move. Note that the game has exactly k moves, that is, throughout the game the coin was tossed exactly k times.

Your task is to determine whether the described situation is possible or else state that Petya was mistaken somewhere. See samples and notes to them for a better understanding.

Input

The first line contains two integers n and k $(1 \leq n, k \leq 100)$. The second line contains n space-separated integers $q_1, q_2, ..., q_n$ $(1 \leq q_i \leq n)$ — the permutation that Petya's got as a present. The third line contains Masha's permutation s, in the similar format.

It is guaranteed that the given sequences q and s are correct permutations.

Output

If the situation that is described in the statement is possible, print "`YES`" (without the quotes), otherwise print "`NO`" (without the quotes).

Sample test(s)

input

```
4 1
2 3 4 1
1 2 3 4
```

output

```
NO
```

input

```
4 1
4 3 1 2
3 4 2 1
```

output

```
YES
```

input

```
4 3
4 3 1 2
3 4 2 1
```

output

```
YES
```

input

```
4 2
4 3 1 2
2 1 4 3
```

output

```
YES
```

input

```
4 1
4 3 1 2
2 1 4 3
```

output

```
NO
```

Note

In the first sample Masha's permutation coincides with the permutation that was written on the board before the beginning of the game. Consequently, that violates the condition that Masha's permutation never occurred on the board before k moves were performed.

In the second sample the described situation is possible, in case if after we toss a coin, we get tails.

In the third sample the possible coin tossing sequence is: heads-tails-tails.

In the fourth sample the possible coin tossing sequence is: heads-heads.

E. Number Transformation

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

Little Petya likes positive integers a lot. Recently his mom has presented him a positive integer a. There's only one thing Petya likes more than numbers: playing with little Masha. It turned out that Masha already has a positive integer b. Petya decided to turn his number a into the number b consecutively performing the operations of the following two types:

1. Subtract 1 from his number.
2. Choose any integer x from 2 to k, inclusive. Then subtract number $(a \ mod \ x)$ from his number a. Operation $a \ mod \ x$ means taking the remainder from division of number a by number x.

Petya performs one operation per second. Each time he chooses an operation to perform during the current move, no matter what kind of operations he has performed by that moment. In particular, this implies that he can perform the same operation any number of times in a row.

Now he wonders in what minimum number of seconds he could transform his number a into number b. Please note that numbers x in the operations of the second type are selected anew each time, independently of each other.

Input

The only line contains three integers a, b $(1 \le b \le a \le 10^{18})$ and k $(2 \le k \le 15)$.

Please do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Output

Print a single integer — the required minimum number of seconds needed to transform number a into number b.

Sample test(s)

input

```
10 1 4
```

output

```
6
```

input

```
6 3 10
```

output

```
2
```

input

```
1000000000000000000 1 3
```

output

```
666666666666666667
```

Note

In the first sample the sequence of numbers that Petya gets as he tries to obtain number b is as follows: 10 $\longrightarrow$ 8 $\longrightarrow$ 6 $\longrightarrow$ 4 $\longrightarrow$ 3 $\longrightarrow$ 2 $\longrightarrow$ 1.

In the second sample one of the possible sequences is as follows: 6 $\longrightarrow$ 4 $\longrightarrow$ 3.