Task 3: Password (`password`)

The elusive Phantom Thief Lupin the Fourth (usually addressed as just Lupin) has taken on the daunting task of breaking into the vault of the largest, most majestic, most technologically advanced castle in the world: The Castle of Gagliostro. Equipped with state of the art technology, it goes without saying that the castle has top-class security which fully utilises the capabilities of man and machine. Nevertheless, Lupin was unfazed even when faced when such a challenge. He stealthily evaded the guards and security cameras positioned around the castle, fought through numerous traps, and solved many puzzles before finally arriving at the vault, where he hacked the security cameras to show a feed that is not at all suspicious and drugged the security guards to render all of them unconscious.

The final obstacle standing in his way is an electronic password system which will open the vault once the correct password is entered. Initially, N non-negative integers between 0 and K inclusive are shown on the screen, the i^{th} of which is equal to A_i. The password also consists of N numbers, each of which is also a non-negative integer between 0 and K inclusive. Through his sheer, blinding brilliance in both mind and body, Lupin determined that the i^{th} number in the password is P_i for each $1 \leq i \leq N$. Upon figuring out the password, Lupin was elated and all ready to complete his mission. To his dismay, however, the way the password is input is rather peculiar, and very time consuming, involving operations which alter the numbers shown on the screen slowly but surely.

Suppose the i^{th} number currently shown on the screen is C_i ($1 \leq i \leq N$). Lupin can open the vault when $C_i = P_i$ for all $1 \leq i \leq n$. Otherwise, Lupin is allowed operations of the following form: choose two integers i, j with $1 \leq i \leq j \leq N$, and for all integers l with $i \leq l \leq j$, C_l changes to $C_l + 1$ (mod $K + 1$). In other words, change C_l to the remainder of $C_l + 1$ when it is divided by $K + 1$. Specifically, 0 changes to 1, 1 changes to 2, ..., $K - 1$ changes to K and K changes to 0.

As if the terrible password input system wasn't bad enough, Lupin has also received information that his archnemesis Inspector Zenigada is rushing over to catch him once and for all. In order to break into the vault and escape as quickly as possible, Lupin wants to use the minimum number of operations needed to change the initial numbers to become the password and open the vault; your task is to determine the minimum number of operations Lupin needs.

Input

Your program must read from standard input.

The first line of the input contains two integers, N and K.

The next line contains N integers $A_1, A_2, \ldots, A_N$, where A_i is the i^{th} number shown on the screen initially.

The third and final line contains N integers $P_1, P_2, \ldots, P_N$, where P_j is the j^{th} number in the password.

Output

Your program must print to standard output.

The output should contain a single integer on a single line, the minimum number of operations Lupin needs to change the numbers on the screen to become the password.

Implementation Note

As the input lengths for some subtasks may be very large, you are recommended to use C++ with fast input routines to solve this problem. The scientific committee does not have a solution written in Java or Python that can fully solve this problem.

C++ and Java source files containing fast input/output templates have been provided in the attachment. You are strongly recommended to use these templates.

If you are implementing your solution in Java, please name your file `Password.java` and place your main function inside `class Password`.

Subtasks

The maximum execution time on each instance is 1.0s, and the maximum memory usage on each instance is 1GiB. For all test-cases, the input will satisfy the following bounds:

- $1 \leq N \leq 3 \times 10^5$
- $0 \leq K \leq 10^9$
- $0 \leq A[i], P[i] \leq K$ for all $1 \leq i \leq N$

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Points	Additional Constraints
1	6	$N \leq 3$
2	5	$A_i \leq A_{i+1}$ for all $1 \leq i \leq N-1$ $P_i = 0$ for all $1 \leq i \leq N$
3	9	$K \leq 1$
4	10	$N, K \leq 80$
5	13	$N \leq 400$
6	23	$N \leq 3000$
7	34	-

Sample Testcase 1

This testcase is valid for subtasks 1, 4, 5, 6, and 7.

Input	Output
3 4 1 2 0 2 1 2	4

Sample Testcase 1 Explanation

An optimal sequence of operations is to choose $(i, j) = (1, 3)$ once, $(2, 3)$ once and $(2, 2)$ twice.

Sample Testcase 2

This testcase is valid for subtasks 3, 4, 5, 6, and 7.

Input	Output
7 1 1 0 1 0 1 1 1 0 0 1 1 0 1 0	3

Sample Testcase 2 Explanation

An optimal sequence of operations is to choose $(i, j) = (1, 3)$ once, $(2, 6)$ once and $(6, 7)$ once.

Sample Testcase 3

This testcase is valid for subtasks 4, 5, 6, and 7.

Input	Output
7 9 1 5 3 4 8 3 2 7 4 8 3 2 3 1	18