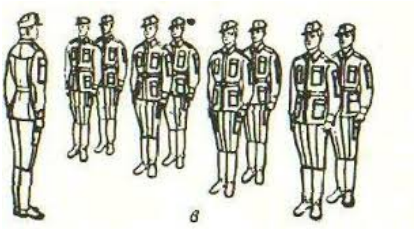
Lines

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Aibar is the commander of the 31-st company. The company soldiers are standing in two lines — the front and the rear lines. In either line, there is the same number of soldiers who are standing in the non-decreasing order of height.



The proper formation is the one in which each soldier standing in the front line is shorter than the soldier standing right behind him in the rear line. To achieve such formation, Aibar came up with a tricky algorithm.

He picks a pair of soldiers on the same position in their according lines, in which the soldier from the front line is higher than his counterpart in the rear line. If there are many such pairs he picks the one closest to the beginning. He orders the two picked soldiers to swap their positions, and then each of the lines is reshuffled back in the non-decreasing order of height. These actions are repeated until the proper formation is achieved.

Aibar wants to know beforehand how many pairs of soldiers will be ordered to swap their positions.

Input

The first line contains a single integer N ($1 \le N \le 10^6$) — the number of soldiers in each line. The next two lines contain the sequences A and B ($1 \le A_i, B_i \le N$) — the heights of the soldiers in the front and the rear lines respectively.

It is guaranteed that initially, soldiers in each line are standing in the non-decreasing order of height.

Output

Output a single integer — the number of swap orders needed to achieve the proper formation.

Scoring

This problem is made up of 5 subtasks, that meet the following constraints:

1. $N \le 1000$. Worth 16 points.
2. $N \le 8000$. Worth 13 points.
3. $A_i, B_i \le 300$, $N \le 3 \times 10^5$. Worth 20 points.
4. $N \le 3 \times 10^5$. Worth 43 points.
5. Only constraints from the statement. Worth 8 points.

Example

standard input	standard output
3 2 2 3 1 1 2	2

Annoying Donchik

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Batyr was given two 2-dimensional arrays A and B of the same size $N \times M$ filled with lowercase letters as a gift for the New Year's Eve. Batyr was disappointed that the arrays were different and decided to fix that. For that, he picks any row or column of the matrix A and reverses it. But Donchik screws everything up: any time Batyr reverses i-th row, Donchik reverses the $N - i + 1$-th row, and whenever Batyr reverses j-th column, Donchik reverses the $M - j + 1$-th column. Donchik does operations on matrix A. Batyr is fed up with Donchik's annoying tricks and he asks for your help. Find the set of operations that Batyr should perform, so that both arrays become identical, or tell that it is impossible.

Input

The first line of the input contains three non-negative integers N, M and *type* ($N \le M$, $1 \le N \times M \le 10^6$, $0 \le type \le 1$) — the sides of the 2-dimensional arrays and the type of the current testcase. The next N lines contain M lowercase letters without whitespaces describing the 2-dimensional array A. The further next N lines describe the array B.

Output

Output "-1" (without quotation marks) if the answer doesn't exist. Otherwise, the first line of the output should contain the number K: for $type = 0$ K can be an arbitrary integer up to 2×10^6, but for $type = 1$ K should be equal to the **minimal** number of operations. In the next K lines output the description of the operations that Batyr should perform. Each operation is given by a character and an integer — character "R" is used to describe the reverse of a row, and "C" is used for columns, while the number describes the index of the reversed row or column. The rows are numbered from 1 to N top-down and the columns are numbered from 1 to M from left to right.

Scoring

This problem is made up of 7 subtasks, that meet the following constraints:

1. $N = 2$, $M = 2$, $type = 1$. Worth 8 points.
2. $N \times M \le 10$, $type = 1$. Worth 13 points.
3. $N = 2$, $type = 1$. Worth 9 points.
4. $3 \le N$, $N \times M \le 100$, $type = 1$. Worth 7 points.
5. $N \times M \le 1000$, $type = 1$. Worth 13 points.
6. $type = 0$. Worth 23 points.
7. $type = 1$. Worth 27 points.

Example

standard input	standard output
2 2 1 ab ba ba ab	1 C 1

Nasty Donchik

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 seconds
Memory limit:	256 megabytes

Batyr had his birthday recently and Tima sent him a unique gift - a sequence A of N integers. Batyr enjoyed the gift very much and immediately started analyzing the properties of the given sequence. Most of all, he is interested in *beuatiful triplets*: triplets (i, j, k) so that $1 \leq i \leq j < k \leq n$ and each number present in the segment of the sequence from i to j (both ends inclusive) is also present in the segment from $j + 1$ to k and vice versa(look at the 'Notes' section for further clarification).

Batyr had been meticulously counting the number of such triplets, but nasty Donchik deleted all his records. Help Batyr — count the number of *beuatiful triplets* using his sequence.

Input

The first line of the input contains a single integer N — the size of Batyr's sequence. The next line contains N integers, i-th describing the i-th element of the sequence a_i $(1 \leq a_i \leq N)$.

Output

Output a single integer — the number of *beatiful triplets*.

Scoring

This task is made up of 7 subtasks, that meet the following constraints:

1. $N \leq 100$. Worth 5 points.
2. $N \leq 500$. Worth 7 points.
3. $N \leq 5000$. Worth 12 points.
4. $N \leq 2 \times 10^5$, $1 \leq a_i \leq 2$, for each $1 \leq i \leq N$. Worth 12 points.
5. $N \leq 2 \times 10^5$, $1 \leq a_i \leq 50$, for each $1 \leq i \leq N$. Worth 12 points.
6. $N \leq 2 \times 10^5$, each number in A is present exactly twice. Worth 12 points.
7. $N \leq 2 \times 10^5$. Worth 40 points.

Examples

standard input	standard output
5 1 2 2 2 1	6
10 1 2 1 1 2 1 2 2 2 1	66

Note

The answer to the first test is the following triplets:

1. (1, 2, 5)
2. (1, 3, 5)
3. (2, 2, 3)
4. (2, 2, 4)
5. (2, 3, 4)
6. (3, 3, 4)