Codeforces Round #543 (Div. 2, based on Technocup 2019 Final Round)

A. Technogoblet of Fire

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Everybody knows that the m-coder Tournament will happen soon. m schools participate in the tournament, and only one student from each school participates.

There are a total of n students in those schools. Before the tournament, all students put their names and the names of their schools into the Technogoblet of Fire. After that, Technogoblet selects the strongest student from each school to participate.

Arkady is a hacker who wants to have k Chosen Ones selected by the Technogoblet. Unfortunately, not all of them are the strongest in their schools, but Arkady can make up some new school names and replace some names from Technogoblet with those. You can't use each made-up name more than once. In that case, Technogoblet would select the strongest student in those made-up schools too.

You know the power of each student and schools they study in. Calculate the minimal number of schools Arkady has to make up so that k Chosen Ones would be selected by the Technogoblet.

Input

The first line contains three integers n, m and k ($1 \le n \le 100$, $1 \le m, k \le n$) — the total number of students, the number of schools and the number of the Chosen Ones.

The second line contains n **different** integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$), where p_i denotes the power of i-th student. The bigger the power, the stronger the student.

The third line contains n integers $s_1, s_2, \ldots, s_n$ ($1 \le s_i \le m$), where s_i denotes the school the i-th student goes to. At least one student studies in each of the schools.

The fourth line contains k **different** integers $c_1, c_2, \ldots, c_k$ ($1 \le c_i \le n$) — the id's of the Chosen Ones.

Output

Output a single integer — the minimal number of schools to be made up by Arkady so that k Chosen Ones would be selected by the Technogoblet.

Examples

input

```
7 3 1
1 5 3 4 6 7 2
1 3 1 2 1 2 3
3
```

output

```
1
```

input

```
8 4 4
1 2 3 4 5 6 7 8
4 3 2 1 4 3 2 1
3 4 5 6
```

output

```
2
```

Note

In the first example there's just a single Chosen One with id 3. His power is equal to 3, but in the same school 1, there's a student with id 5 and power 6, and that means inaction would not lead to the latter being chosen. If we, however, make up a new school (let its id be 4) for the Chosen One, Technogoblet would select students with ids 2 (strongest in 3), 5 (strongest in 1), 6 (strongest in 2) and 3 (strongest in 4).

In the second example, you can change the school of student 3 to the made-up 5 and the school of student 4 to the made-up 6. It will cause the Technogoblet to choose students 8, 7, 6, 5, 3 and 4.

B. Mike and Children

time limit per test: 2 seconds
memory limit per test: 256 megabytes

input: standard input
output: standard output

Mike decided to teach programming to children in an elementary school. He knows that it is not an easy task to interest children in that age to code. That is why he decided to give each child **two** sweets.

Mike has n sweets with sizes $a_1, a_2, \ldots, a_n$. All his sweets have **different** sizes. That is, there is no such pair (i, j) ($1 \leq i, j \leq n$) such that $i \neq j$ and $a_i = a_j$.

Since Mike has taught for many years, he knows that if he gives two sweets with sizes a_i and a_j to one child and a_k and a_p to another, where $(a_i + a_j) \neq (a_k + a_p)$, then a child who has a smaller sum of sizes will be upset. That is, if there are two children who have different sums of sweets, then one of them will be upset. Apparently, Mike does not want somebody to be upset.

Mike wants to invite children giving each of them **two** sweets. Obviously, he can't give one sweet to two or more children. His goal is to invite as many children as he can.

Since Mike is busy preparing to his first lecture in the elementary school, he is asking you to find the maximum number of children he can invite giving each of them two sweets in such way that nobody will be upset.

Input

The first line contains one integer n ($2 \leq n \leq 1\,000$) — the number of sweets Mike has.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^5$) — the sizes of the sweets. It is guaranteed that all integers are distinct.

Output

Print one integer — the maximum number of children Mike can invite giving each of them two sweets in such way that nobody will be upset.

Examples

input
```
8
1 8 3 11 4 9 2 7
```
output
```
3
```

input
```
7
3 1 7 11 9 2 12
```
output
```
2
```

Note

In the first example, Mike can give $9 + 2 = 11$ to one child, $8 + 3 = 11$ to another one, and $7 + 4 = 11$ to the third child. Therefore, Mike can invite three children. Note that it is **not** the only solution.

In the second example, Mike can give $3 + 9 = 12$ to one child and $1 + 11$ to another one. Therefore, Mike can invite two children. Note that it is **not** the only solution.

C. System Testing

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Vasya likes taking part in Codeforces contests. When a round is over, Vasya follows all submissions in the system testing tab.

There are n solutions, the i-th of them should be tested on a_i tests, testing one solution on one test takes 1 second. The solutions are judged in the order from 1 to n. There are k testing processes which test solutions simultaneously. Each of them can test at most one solution at a time.

At any time moment t when some testing process is not judging any solution, it takes the first solution from the queue and tests it on each test in increasing order of the test ids. Let this solution have id i, then it is being tested on the first test from time moment t till time moment $t + 1$, then on the second test till time moment $t + 2$ and so on. This solution is fully tested at time moment $t + a_i$, and after that the testing process immediately starts testing another solution.

Consider some time moment, let there be exactly m fully tested solutions by this moment. There is a caption "System testing: d%" on the page with solutions, where d is calculated as

$$d = round\left(100 \cdot \frac{m}{n}\right),$$

where $round(x) = \lfloor x + 0.5 \rfloor$ is a function which maps every real to the nearest integer.

Vasya calls a submission *interesting* if there is a time moment (possibly, non-integer) when the solution is being tested on some test q, and the caption says "System testing: q%". Find the number of interesting solutions.

Please note that in case when multiple processes attempt to take the first submission from the queue at the same moment (for instance, at the initial moment), the order they take the solutions does not matter.

Input

The first line contains two positive integers n and k ($1 \le n \le 1000$, $1 \le k \le 100$) standing for the number of submissions and the number of testing processes respectively.

The second line contains n positive integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 150$), where a_i is equal to the number of tests the i-th submission is to be run on.

Output

Output the only integer — the number of interesting submissions.

Examples

input

```
2 2
49 100
```

output

```
1
```

input

```
4 2
32 100 33 1
```

output

```
2
```

input

```
14 5
48 19 6 9 50 20 3 42 38 43 36 21 44 6
```

output

```
5
```

Note

Consider the first example. At time moment 0 both solutions start testing. At time moment 49 the first solution is fully tested, so at time moment 49.5 the second solution is being tested on the test 50, and the caption says "System testing: 50%" (because there is one fully tested solution out of two). So, the second solution is interesting.

Consider the second example. At time moment 0 the first and the second solutions start testing. At time moment 32 the first solution is fully tested, the third solution starts testing, the caption says "System testing: 25%". At time moment $32 + 24.5 = 56.5$ the third solutions is being tested on test 25, the caption is still the same, thus this solution is interesting. After that the third solution is fully tested at time moment $32 + 33 = 65$, the fourth solution is fully tested at time moment $65 + 1 = 66$. The captions becomes "System testing: 75%", and at time moment 74.5 the second solution is being tested on test 75. So, this solution is also interesting. Overall, there are two interesting solutions.

D. Diana and Liana

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

At the first holiday in spring, the town Shortriver traditionally conducts a flower festival. Townsfolk wear traditional wreaths during these festivals. Each wreath contains exactly k flowers.

The work material for the wreaths for all n citizens of Shortriver is cut from the longest flowered liana that grew in the town that year. Liana is a sequence a_1, a_2, ..., a_m, where a_i is an integer that denotes the type of flower at the position i. This year the liana is very long ($m \ge n \cdot k$), and that means every citizen will get a wreath.

Very soon the liana will be inserted into a special cutting machine in order to make work material for wreaths. The machine works in a simple manner: it cuts k flowers from the beginning of the liana, then another k flowers and so on. Each such piece of k flowers is called a workpiece. The machine works until there are less than k flowers on the liana.

Diana has found a weaving schematic for the most beautiful wreath imaginable. In order to weave it, k flowers must contain flowers of types b_1, b_2, ..., b_s, while other can be of any type. If a type appears in this sequence several times, there should be at least that many flowers of that type as the number of occurrences of this flower in the sequence. The order of the flowers in a workpiece does not matter.

Diana has a chance to remove some flowers from the liana before it is inserted into the cutting machine. She can remove flowers from any part of the liana without breaking liana into pieces. If Diana removes too many flowers, it may happen so that some of the citizens do not get a wreath. Could some flowers be removed from the liana so that at least one workpiece would conform to the schematic and machine would still be able to create at least n workpieces?

Input

The first line contains four integers m, k, n and s ($1 \leq n, k, m \leq 5 \cdot 10^5$, $k \cdot n \leq m$, $1 \leq s \leq k$): the number of flowers on the liana, the number of flowers in one wreath, the amount of citizens and the length of Diana's flower sequence respectively.

The second line contains m integers a_1, a_2, ..., a_m ($1 \leq a_i \leq 5 \cdot 10^5$) — types of flowers on the liana.

The third line contains s integers b_1, b_2, ..., b_s ($1 \leq b_i \leq 5 \cdot 10^5$) — the sequence in Diana's schematic.

Output

If it's impossible to remove some of the flowers so that there would be at least n workpieces and at least one of them fullfills Diana's schematic requirements, output -1.

Otherwise in the first line output one integer d — the number of flowers to be removed by Diana.

In the next line output d different integers — the positions of the flowers to be removed.

If there are multiple answers, print any.

Examples

input

```
7 3 2 2
1 2 3 3 2 1 2
2 2
```

output

```
1
4
```

input

```
13 4 3 3
3 2 6 4 1 4 4 7 1 3 3 2 4
4 3 4
```

output

```
-1
```

input

```
13 4 1 3
3 2 6 4 1 4 4 7 1 3 3 2 4
4 3 4
```

output

```
9
1 2 3 4 5 9 11 12 13
```

Note

In the first example, if you don't remove any flowers, the machine would put out two workpieces with flower types $[1, 2, 3]$ and $[3, 2, 1]$. Those workpieces don't fit Diana's schematic. But if you remove flower on 4-th place, the machine would output workpieces $[1, 2, 3]$ and $[2, 1, 2]$. The second workpiece fits Diana's schematic.

In the second example there is no way to remove flowers so that every citizen gets a wreath and Diana gets a workpiece that fits here schematic.

In the third example Diana is the only citizen of the town and that means she can, for example, just remove all flowers except the ones she needs.

E. Once in a casino

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

One player came to a casino and found a slot machine where everything depends only on how he plays. The rules follow.

A positive integer a is initially on the screen. The player can put a coin into the machine and then add 1 to or subtract 1 from any two adjacent digits. All digits must remain from 0 to 9 after this operation, and the leading digit must not equal zero. In other words, it is forbidden to add 1 to 9, to subtract 1 from 0 and to subtract 1 from the leading 1. Once the number on the screen becomes equal to b, the player wins the jackpot. a and b have the same number of digits.

Help the player to determine the minimal number of coins he needs to spend in order to win the jackpot and tell how to play.

Input

The first line contains a single integer n ($2 \leq n \leq 10^5$) standing for the length of numbers a and b.

The next two lines contain numbers a and b, each one on a separate line ($10^{n-1} \le a, b < 10^n$).

Output

If it is impossible to win the jackpot, print a single integer -1.

Otherwise, the first line must contain the minimal possible number c of coins the player has to spend.

$\min(c, 10^5)$ lines should follow, i-th of them containing two integers d_i and s_i ($1 \le d_i \le n - 1$, $s_i = \pm 1$) denoting that on the i-th step the player should add s_i to the d_i-th and $(d_i + 1)$-st digits from the left (e. g. $d_i = 1$ means that two leading digits change while $d_i = n - 1$ means that there are two trailing digits which change).

Please notice that the answer may be very big and in case $c > 10^5$ you should print only the first 10^5 moves. Your answer is considered correct if it is possible to finish your printed moves to win the jackpot in the minimal possible number of coins. In particular, if there are multiple ways to do this, you can output any of them.

Examples

input

```
3
223
322
```

output

```
2
1 1
2 -1
```

input

```
2
20
42
```

output

```
2
1 1
1 1
```

input

```
2
35
44
```

output

```
-1
```

Note

In the first example, we can make a +1 operation on the two first digits, transforming number $\mathbf{22}3$ into $\mathbf{33}3$, and then make a -1 operation on the last two digits, transforming $3\mathbf{33}$ into $3\mathbf{22}$.

It's also possible to do these operations in reverse order, which makes another correct answer.

In the last example, one can show that it's impossible to transform 35 into 44.

F. Compress String

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Suppose you are given a string s of length n consisting of lowercase English letters. You need to compress it using the smallest possible number of coins.

To compress the string, you have to represent s as a concatenation of several non-empty strings: $s = t_1 t_2 \ldots t_k$. The i-th of these strings should be encoded with one of the two ways:

- if $|t_i| = 1$, meaning that the current string consists of a single character, you can encode it paying a coins;
- if t_i is a substring of $t_1 t_2 \ldots t_{i-1}$, then you can encode it paying b coins.

A string x is a substring of a string y if x can be obtained from y by deletion of several (possibly, zero or all) characters from the beginning and several (possibly, zero or all) characters from the end.

So your task is to calculate the minimum possible number of coins you need to spend in order to compress the given string s.

Input

The first line contains three positive integers, separated by spaces: n, a and b ($1 \le n, a, b \le 5000$) — the length of the string, the cost to compress a one-character string and the cost to compress a string that appeared before.

The second line contains a single string s, consisting of n lowercase English letters.

Output

Output a single integer — the smallest possible number of coins you need to spend to compress s.

Examples

input

```
3 3 1
aba
```

output

```
7
```

input

```
4 1 1
abcd
```

output

```
4
```

input

```
4 10 1
aaaa
```

output

```
12
```

Note

In the first sample case, you can set $t_1 =$ 'a', $t_2 =$ 'b', $t_3 =$ 'a' and pay $3 + 3 + 1 = 7$ coins, since t_3 is a substring of $t_1 t_2$.

In the second sample, you just need to compress every character by itself.

In the third sample, you set $t_1 = t_2 =$ 'a', $t_3 =$ 'aa' and pay $10 + 1 + 1 = 12$ coins, since t_2 is a substring of t_1 and t_3 is a substring of $t_1 t_2$.