Codeforces Round #122 (Div. 1)

A. Cutting Figure

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

You've gotten an $n \times m$ sheet of squared paper. Some of its squares are painted. Let's mark the set of all painted squares as A. Set A is connected. Your task is to find the minimum number of squares that we can delete from set A to make it not connected.

A set of painted squares is called *connected*, if for every two squares a and b from this set there is a sequence of squares from the set, beginning in a and ending in b, such that in this sequence any square, except for the last one, shares a common side with the square that follows next in the sequence. An empty set and a set consisting of exactly one square are connected by definition.

Input

The first input line contains two space-separated integers n and m ($1 \le n, m \le 50$) — the sizes of the sheet of paper.

Each of the next n lines contains m characters — the description of the sheet of paper: the j-th character of the i-th line equals either "#", if the corresponding square is painted (belongs to set A), or equals "." if the corresponding square is not painted (does not belong to set A). It is guaranteed that the set of all painted squares A is connected and isn't empty.

Output

On the first line print the minimum number of squares that need to be deleted to make set A not connected. If it is impossible, print -1.

Examples

input
5 4 #### #..# #..# #..# ####
output
2

input
5 5 ##### #...# ##### #...# #####
output
2

Note

In the first sample you can delete any two squares that do not share a side. After that the set of painted squares is not connected anymore.

The note to the second sample is shown on the figure below. To the left there is a picture of the initial set of squares. To the right there is a set with deleted squares. The deleted squares are marked with crosses.

B. Xor

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

John Doe has four arrays: a, b, k, and p. Each array consists of n integers. Elements of all arrays are indexed starting from 1. Array p is a permutation of integers 1 to n.

John invented a game for his friends and himself. Initially a player is given array a. The player must consecutively execute exactly u operations on a. You are permitted to execute the following operations:

- Operation 1: For each $i \in \{1, 2, \ldots, n\}$ change a_i into $a_i \oplus b_i$. Expression $x \oplus y$ means applying the operation of a bitwise xor to numbers x and y. The given operation exists in all modern programming languages, for example, in language *C++* and *Java* it is marked as "^", in *Pascal* — as "xor".
- Operation 2: For each $i \in \{1, 2, \ldots, n\}$ change a_i into $a_{p_i} + r$. When this operation is executed, all changes are made at the same time.

After all u operations are applied, the number of points the player gets is determined by the formula $s = \sum_{i=1}^{i \le n} a_i k_i$.

John wants to find out what maximum number of points a player can win in his game. Help him.

Input

The first line contains space-separated integers n, u and r ($1 \le n, u \le 30$, $0 \le r \le 100$) — the number of elements in each array, the number of operations and the number that describes one of the operations.

Each of the next four lines contains n space-separated integers — arrays a, b, k, p. The first line has array a, the second line has array b, the third line has array k and the fourth one has array p.

It is guaranteed that elements of arrays a and b are positive and do not exceed 10^4 ($1 \le a_i, b_i \le 10^4$), elements of array k do not exceed 10^4 in the absolute value ($|k| \le 10^4$) and p is a permutation of numbers from 1 to n.

Output

On a single line print number s — the maximum number of points that a player can win in John's game.

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.

Examples

input
3 2 1 7 7 7 8 8 8 1 2 3 1 3 2

output
96

input
2 1 0 1 1 1 1 1 -1 1 2

output
0

Note

In the first sample you should first apply the operation of the first type, then the operation of the second type.

C. Hamming Distance

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

Hamming distance between strings a and b of equal length (denoted by $h(a, b)$) is equal to the number of distinct integers i $(1 \le i \le |a|)$, such that $a_i \ne b_i$, where a_i is the i-th symbol of string a, b_i is the i-th symbol of string b. For example, the Hamming distance between strings "aba" and "bba" equals 1, they have different first symbols. For strings "bbba" and "aaab" the Hamming distance equals 4.

John Doe had a paper on which four strings of equal length s_1, s_2, s_3 and s_4 were written. Each string s_i consisted only of lowercase letters "a" and "b". John found the Hamming distances between all pairs of strings he had. Then he lost the paper with the strings but he didn't lose the Hamming distances between all pairs.

Help John restore the strings; find some four strings s'_1, s'_2, s'_3, s'_4 of equal length that consist only of lowercase letters "a" and "b", such that the pairwise Hamming distances between them are the same as between John's strings. More formally, set s'_i must satisfy the condition $\forall i, j \in \{1, 2, 3, 4\}, h(s_i, s_j) = h(s'_i, s'_j)$.

To make the strings easier to put down on a piece of paper, you should choose among all suitable sets of strings the one that has strings of **minimum length**.

Input

The first line contains space-separated integers $h(s_1, s_2)$, $h(s_1, s_3)$, $h(s_1, s_4)$. The second line contains space-separated integers $h(s_2, s_3)$ and $h(s_2, s_4)$. The third line contains the single integer $h(s_3, s_4)$.

All given integers $h(s_i, s_j)$ are non-negative and do not exceed 10^5. It is guaranteed that at least one number $h(s_i, s_j)$ is positive.

Output

Print -1 if there's no suitable set of strings.

Otherwise print on the first line number len — the length of each string. On the i-th of the next four lines print string s'_i. If there are multiple sets with the minimum length of the strings, print any of them.

Examples

input
4 4 4 4 4 4
output
6 aaaabb aabbaa bbaaaa bbbbbb

D. Two Segments

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Nick has some permutation consisting of p integers from 1 to n. A segment $[l, r]$ ($l \le r$) is a set of elements p_i satisfying $l \le i \le r$.

Nick calls a pair of segments $[a_0, a_1]$ and $[b_0, b_1]$ ($1 \le a_0 \le a_1 < b_0 \le b_1 \le n$) good if all their $(a_1 - a_0 + b_1 - b_0 + 2)$ elements, when sorted in ascending order, form an arithmetic progression with a difference of 1. That is, when they sorted in ascending order, the elements are in the form $\{x, x + 1, x + 2, ..., x + m - 1\}$, for some x and m.

Your task is to find the number of distinct pairs of good segments in the given permutation. Two pairs of segments are considered distinct if the sets of elements contained in these pairs of segments are distinct. For example, any segment $[l, r]$ ($l < r$) can be represented as a pair of segments, as $[l, i]$ and $[i + 1, r]$ ($l \le i \le r$). As all these pairs consist of the same set of elements, they are considered identical.

See the notes accompanying the sample tests for clarification.

Input

The first line contains integer n ($1 \le n \le 3 \cdot 10^5$) — the permutation size. The second line contains n space-separated distinct integers p_i, ($1 \le p_i \le n$).

Output

Print a single integer — the number of good pairs of segments of permutation p.

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.

Examples

input
3 1 2 3
output
3

input
5 1 4 5 3 2
output
10

input
5 5 4 3 1 2
output
10

Note

In the first sample the following pairs of segments are good: ($[1, 1]$, $[2, 2]$); ($[2, 2]$, $[3, 3]$); ($[1, 2]$, $[3, 3]$). Pair of segments ($[1, 1]$, $[2, 3]$) is by definition equivalent to pair ($[1, 2]$, $[3, 3]$), since both of them covers the same set of elements, namely $\{1, 2, 3\}$.

In the third sample the following pairs of segments are good: ($[4, 4]$, $[5, 5]$); ($[3, 3]$,$[4, 5]$); ($[2, 2]$,$[3, 5]$); ($[1, 1]$,$[2, 5]$); ($[3, 3]$,$[5, 5]$); ($[2, 3]$,$[5, 5]$); ($[1, 3]$,$[5, 5]$); ($[2, 2]$,$[3, 3]$); ($[1, 1]$,$[2, 3]$); ($[1, 1]$,$[2, 2]$).

E. Fibonacci Number

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

John Doe has a list of all Fibonacci numbers modulo 10^{13}. This list is infinite, it starts with numbers 0 and 1. Each number in the list, apart from the first two, is a sum of previous two modulo 10^{13}. That is, John's list is made from the Fibonacci numbers' list by replacing each number there by the remainder when divided by 10^{13}.

John got interested in number f ($0 \le f < 10^{13}$) and now wants to find its first occurrence in the list given above. Help John and find the number of the first occurence of number f in the list or otherwise state that number f does not occur in the list.

The numeration in John's list starts from zero. There, the 0-th position is the number 0, the 1-st position is the number 1, the 2-nd position is the number 1, the 3-rd position is the number 2, the 4-th position is the number 3 and so on. Thus, the beginning of the list looks like this: $0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots$

Input

The first line contains the single integer f ($0 \le f < 10^{13}$) — the number, which position in the list we should find.

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 number — the number of the first occurrence of the given number in John's list. If this number doesn't occur in John's list, print -1.

Examples

input
13
output
7

input
377
output
14