A. Magical Array

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

Valery is very interested in magic. Magic attracts him so much that he sees it everywhere. He explains any strange and weird phenomenon through intervention of supernatural forces. But who would have thought that even in a regular array of numbers Valera manages to see something beautiful and magical.

Valera absolutely accidentally got a piece of ancient parchment on which an array of numbers was written. He immediately thought that the numbers in this array were not random. As a result of extensive research Valera worked out a wonderful property that a magical array should have: an array is defined as magic if its **minimum and maximum coincide**.

He decided to share this outstanding discovery with you, but he asks you for help in return. Despite the tremendous intelligence and wit, Valera counts very badly and so you will have to complete his work. All you have to do is count the number of magical subarrays of the original array of numbers, written on the parchment. Subarray is defined as **non-empty sequence of consecutive elements**.

Input

The first line of the input data contains an integer n ($1 \le n \le 10^5$). The second line contains an array of original integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$).

Output

Print on the single line the answer to the problem: the amount of subarrays, which are magical.

Please do not use the `%lld` specificator to read or write 64-bit numbers in C++. It is recommended to use `cin`, `cout` streams (you can also use the `%I64d` specificator).

Examples

input

```
4
2 1 1 4
```

output

```
5
```

input

```
5
-2 -2 -2 0 1
```

output

```
8
```

Note

Notes to sample tests:

Magical subarrays are shown with pairs of indices [a;b] of the beginning and the end.

In the first sample: [1;1], [2;2], [3;3], [4;4], [2;3].

In the second sample: [1;1], [2;2], [3;3], [4;4], [5;5], [1;2], [2;3], [1;3].

B. Doctor

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

There are n animals in the queue to Dr. Dolittle. When an animal comes into the office, the doctor examines him, gives prescriptions, appoints tests and may appoint extra examination. Doc knows all the forest animals perfectly well and therefore knows exactly that the animal number i in the queue will have to visit his office exactly a_i times. We will assume that an examination takes much more time than making tests and other extra procedures, and therefore we will assume that once an animal leaves the room, it immediately gets to the end of the queue to the doctor. Of course, if the animal has visited the doctor as many times as necessary, then it doesn't have to stand at the end of the queue and it immediately goes home.

Doctor plans to go home after receiving k animals, and therefore what the queue will look like at that moment is important for him. Since the doctor works long hours and she can't get distracted like that after all, she asked you to figure it out.

Input

The first line of input data contains two space-separated integers n and k ($1 \le n \le 10^5$, $0 \le k \le 10^{14}$). In the second line are given space-separated integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$).

Please do not use the `%lld` specificator to read or write 64-bit numbers in C++. It is recommended to use `cin`, `cout` streams (you can also use the `%I64d` specificator).

Output

If the doctor will overall carry out less than k examinations, print a single number "-1" (without quotes). Otherwise, print the sequence of numbers — number of animals in the order in which they stand in the queue.

Note that this sequence may be empty. This case is present in pretests. You can just print nothing or print one "End of line"-character. Both will be accepted.

Examples

input

```
3 3
1 2 1
```

output

```
2
```

input

```
4 10
3 3 2 1
```

output

```
-1
```

input

```
7 10
1 3 3 1 2 3 1
```

output

```
6 2 3
```

Note

In the first sample test:

- Before examination: $\{1, 2, 3\}$
- After the first examination: $\{2, 3\}$
- After the second examination: $\{3, 2\}$
- After the third examination: $\{2\}$

In the second sample test:

- Before examination: $\{1, 2, 3, 4, 5, 6, 7\}$
- After the first examination: $\{2, 3, 4, 5, 6, 7\}$
- After the second examination: $\{3, 4, 5, 6, 7, 2\}$
- After the third examination: $\{4, 5, 6, 7, 2, 3\}$
- After the fourth examination: $\{5, 6, 7, 2, 3\}$
- After the fifth examination: $\{6, 7, 2, 3, 5\}$

- After the sixth examination: $\{7, 2, 3, 5, 6\}$
- After the seventh examination: $\{2, 3, 5, 6\}$
- After the eighth examination: $\{3, 5, 6, 2\}$
- After the ninth examination: $\{5, 6, 2, 3\}$
- After the tenth examination: $\{6, 2, 3\}$

C. Track

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

You already know that Valery's favorite sport is biathlon. Due to your help, he learned to shoot without missing, and his skills are unmatched at the shooting range. But now a smaller task is to be performed, he should learn to complete the path fastest.

The track's map is represented by a rectangle $n \times m$ in size divided into squares. Each square is marked with a lowercase Latin letter (which means the type of the plot), with the exception of the starting square (it is marked with a capital Latin letters S) and the terminating square (it is marked with a capital Latin letter T). The time of movement from one square to another is equal to 1 minute. The time of movement within the cell can be neglected. We can move from the cell only to side-adjacent ones, but it is forbidden to go beyond the map edges. Also the following restriction is imposed on the path: it is not allowed to visit more than k **different types** of squares (squares of one type can be visited an infinite number of times). Squares marked with S and T have no type, so they are not counted. But S must be visited exactly once — at the very beginning, and T must be visited exactly once — at the very end.

Your task is to find the path from the square S to the square T that takes minimum time. Among all shortest paths you should choose the **lexicographically minimal** one. When comparing paths you should lexicographically represent them as a sequence of characters, that is, of plot types.

Input

The first input line contains three integers n, m and k ($1 \le n, m \le 50, n \cdot m \ge 2, 1 \le k \le 4$). Then n lines contain the map. Each line has the length of exactly m characters and consists of lowercase Latin letters and characters S and T. It is guaranteed that the map contains exactly one character S and exactly one character T.

Pretest 12 is one of the maximal tests for this problem.

Output

If there is a path that satisfies the condition, print it as a sequence of letters — the plot types. Otherwise, print "-1" (without quotes). **You shouldn't print the character S in the beginning and T in the end**.

Note that this sequence may be empty. This case is present in pretests. You can just print nothing or print one "End of line"-character. Both will be accepted.

Examples

input

```
5 3 2
Sba
ccc
aac
ccc
abT
```

output

```
bcccc
```

input

```
3 4 1
Sxyy
yxxx
yyyT
```

output

```
xxxx
```

input

```
1 3 3
TyS
```

output

```
y
```

input

```
1 4 1
SxyT
```

output

```
-1
```

D. Numbers

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

One quite ordinary day Valera went to school (there's nowhere else he should go on a week day). In a maths lesson his favorite teacher Ms. Evans told students about divisors. Despite the fact that Valera loved math, he didn't find this particular topic interesting. Even more, it seemed so boring that he fell asleep in the middle of a lesson. And only a loud ringing of a school bell could interrupt his sweet dream.

Of course, the valuable material and the teacher's explanations were lost. However, Valera will one way or another have to do the homework. As he does not know the new material absolutely, he cannot do the job himself. That's why he asked you to help. You're his best friend after all, you just cannot refuse to help.

Valera's home task has only one problem, which, though formulated in a very simple way, has not a trivial solution. Its statement looks as follows: if we consider all positive integers in the interval $[a;b]$ then it is required to count the amount of such numbers in this interval that their **smallest divisor** will be a certain integer k **(you do not have to consider divisor equal to one)**. In other words, you should count the amount of such numbers from the interval $[a;b]$, that are not divisible by any number between 2 and $k - 1$ and yet are divisible by k.

Input

The first and only line contains three positive integers a, b, k ($1 \le a \le b \le 2 \cdot 10^9$, $2 \le k \le 2 \cdot 10^9$).

Output

Print on a single line the answer to the given problem.

Examples

input
```
1 10 2
```
output
```
5
```

input
```
12 23 3
```
output
```
2
```

input
```
6 19 5
```
output
```
0
```

Note

Comments to the samples from the statement:

In the first sample the answer is numbers $2, 4, 6, 8, 10$.

In the second one — $15, 21$

In the third one there are no such numbers.

E. Two Subsequences

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

On an IT lesson Valera studied data compression. The teacher told about a new method, which we shall now describe to you.

Let $\{a_1, a_2, \dots, a_n\}$ be the given sequence of lines needed to be compressed. Here and below we shall assume that all lines are **of the same length** and consist only of the digits 0 and 1. Let's define the compression function:

- f(empty sequence) $=$ empty string
- $f(s) = s$.
- $f(s_1, s_2) =$ the smallest in length string, which has one of the prefixes equal to s_1 and one of the suffixes equal to s_2. For example, $f(001, 011) = 0011$, $f(111, 011) = 111011$.
- $f(a_1, a_2, \dots, a_n) = f(f(a_1, a_2, a_{n-1}), a_n)$. For example, $f(000, 000, 111) = f(f(000, 000), 111) = f(000, 111) = 000111$.

Valera faces a real challenge: he should divide the given sequence $\{a_1, a_2, \dots, a_n\}$ into two subsequences $\{b_1, b_2, \dots, b_k\}$ and $\{c_1, c_2, \dots, c_m\}$, $m + k = n$, so that the value of $S = |f(b_1, b_2, \dots, b_k)| + |f(c_1, c_2, \dots, c_m)|$ took the minimum possible value. Here $|p|$ denotes the length of the string p.

Note that it is not allowed to change the relative order of lines in the subsequences. It is allowed to make one of the subsequences empty. Each string from the initial sequence should belong to exactly one subsequence. Elements of subsequences b and c don't have to be consecutive in the original sequence a, i. e. elements of b and c can alternate in a (see samples 2 and 3).

Help Valera to find the minimum possible value of S.

Input

The first line of input data contains an integer n — the number of strings ($1 \le n \le 2 \cdot 10^5$). Then on n lines follow elements of the sequence — strings whose lengths are from 1 to 20 characters, consisting only of digits 0 and 1. The $i + 1$-th input line contains the i-th element of the sequence. Elements of the sequence are separated only by a newline. It is guaranteed that all lines have the same length.

Output

Print a single number — the minimum possible value of S.

Examples

input

```
3
01
10
01
```

output

```
4
```

input

```
4
000
111
110
001
```

output

```
8
```

input

```
5
10101
01010
11111
01000
10010
```

output

```
17
```

Note

Detailed answers to the tests:

- The best option is to make one of the subsequences empty, and the second one equal to the whole given sequence. $|f(01,10,01)| = |f(f(01,10),01)| = |f(010,01)| = |0101| = 4$.
- The best option is: $b = \{000,001\}, c = \{111,110\}$. $S = |f(000,001)| + |f(111,110)| = |0001| + |1110| = 8$.
- The best option is: $b = \{10101,01010,01000\}, c = \{11111,10010\}$. $S = |10101000| + |111110010| = 17$.