Codeforces Beta Round #40 (Div. 2)

A. Translation

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

The translation from the Berland language into the Birland language is not an easy task. Those languages are very similar: a berlandish word differs from a birlandish word with the same meaning a little: it is spelled (and pronounced) reversely. For example, a Berlandish word `code` corresponds to a Birlandish word `edoc`. However, it's easy to make a mistake during the «translation». Vasya translated word s from Berlandish into Birlandish as t. Help him: find out if he translated the word correctly.

Input

The first line contains word s, the second line contains word t. The words consist of lowercase Latin letters. The input data do not consist unnecessary spaces. The words are not empty and their lengths do not exceed 100 symbols.

Output

If the word t is a word s, written reversely, print `YES`, otherwise print `NO`.

Sample test(s)

input
code edoc
output
YES

input
abb aba
output
NO

input
code code
output
NO

B. Martian Dollar

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

One day Vasya got hold of information on the Martian dollar course in bourles for the next n days. The buying prices and the selling prices for one dollar on day i are the same and are equal to a_i. Vasya has b bourles. He can buy a certain number of dollars and then sell it no more than once in n days. According to Martian laws, one can buy only an integer number of dollars. Which maximal sum of money in bourles can Vasya get by the end of day n?

Input

The first line contains two integers n and b $(1 \le n, b \le 2000)$ — the number of days and the initial number of money in bourles. The next line contains n integers a_i $(1 \le a_i \le 2000)$ — the prices of Martian dollars.

Output

Print the single number — which maximal sum of money in bourles can Vasya get by the end of day n.

Sample test(s)

input
2 4 3 7
output
8

input
4 10 4 3 2 1
output
10

input
4 10 4 2 3 1
output
15

C. Email address

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

Sometimes one has to spell email addresses over the phone. Then one usually pronounces a dot as `dot`, an at sign as `at`. As a result, we get something like `vasyaatgmaildotcom`. Your task is to transform it into a proper email address (vasya@gmail.com).

It is known that a proper email address contains only such symbols as `.` `@` and lower-case Latin letters, doesn't start with and doesn't end with a dot. Also, a proper email address doesn't start with and doesn't end with an at sign. Moreover, an email address contains exactly one such symbol as `@`, yet may contain any number (possible, zero) of dots.

You have to carry out a series of replacements so that the length of the result was as short as possible and it was a proper email address. If the lengths are equal, you should print the lexicographically minimal result.

Overall, two variants of replacement are possible: `dot` can be replaced by a dot, `at` can be replaced by an at.

Input

The first line contains the email address description. It is guaranteed that that is a proper email address with all the dots replaced by `dot` an the at signs replaced by `at`. The line is not empty and its length does not exceed 100 symbols.

Output

Print the shortest email address, from which the given line could be made by the described above replacements. If there are several solutions to that problem, print the lexicographically minimal one (the lexicographical comparison of the lines are implemented with an operator < in modern programming languages).

In the ASCII table the symbols go in this order: `.` `@` `ab...z`

Sample test(s)

input
vasyaatgmaildotcom
output
vasya@gmail.com

input
dotdotdotatdotdotat
output
dot..@..at

input
aatt
output
a@t

D. Pawn

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

On some square in the lowest row of a chessboard a stands a pawn. It has only two variants of moving: upwards and leftwards or upwards and rightwards. The pawn can choose from which square of the lowest row it can start its journey. On each square lay from 0 to 9 peas. The pawn wants to reach the uppermost row having collected as many peas as possible. As there it will have to divide the peas between itself and its k brothers, the number of peas must be divisible by $k + 1$. Find the maximal number of peas it will be able to collect and which moves it should make to do it.

The pawn cannot throw peas away or leave the board. When a pawn appears in some square of the board (including the first and last square of the way), it necessarily takes all the peas.

Input

The first line contains three integers n, m, k ($2 \le n, m \le 100, 0 \le k \le 10$) — the number of rows and columns on the chessboard, the number of the pawn's brothers. Then follow n lines containing each m numbers from 0 to 9 without spaces — the chessboard's description. Each square is described by one number — the number of peas in it. The first line corresponds to the uppermost row and the last line — to the lowest row.

Output

If it is impossible to reach the highest row having collected the number of peas divisible by $k + 1$, print `-1`.

Otherwise, the first line must contain a single number — the maximal number of peas the pawn can collect given that the number must be divisible by $k + 1$. The second line must contain a single number — the number of the square's column in the lowest row, from which the pawn must start its journey. The columns are numbered from the left to the right with integral numbers starting from 1. The third line must contain a line consisting of n - 1 symbols — the description of the pawn's moves. If the pawn must move upwards and leftwards, print `L`, if it must move upwards and rightwards, print `R`. If there are several solutions to that problem, print any of them.

Sample test(s)

input

```
3 3 1
123
456
789
```

output

```
16
2
RL
```

input

```
3 3 0
123
456
789
```

output

```
17
3
LR
```

input

```
2 2 10
98
75
```

output

```
-1
```

E. 3-cycles

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

During a recent research Berland scientists found out that there were n cities in Ancient Berland, joined by two-way paths. Any two cities are joined by no more than one path. No path joins a city with itself. According to a well-known tradition, the road network was built so that it would be impossible to choose three cities from each of which one can get to any other one directly. That is, there was no cycle exactly as long as 3. Unfortunately, the road map has not been preserved till nowadays. Now the scientists are interested how much developed a country Ancient Berland was. Help them - find, what maximal number of roads could be in the country. You also have to restore any of the possible road maps.

Input

The first line contains an integer n $(1 \leq n \leq 100)$ — the number of cities in Berland.

Output

On the first line must be printed number m — the maximal number of roads in Berland. Then print m lines containing two numbers each — the numbers of cities that the given road joins. The cities are numbered with integers from 1 to n. If there are several variants of solving the problem, print any of them.

Sample test(s)

input
3

output
2 1 2 2 3

input
4

output
4 1 2 2 3 3 4 4 1