



Sequence of Numbers

Task ID: sequence	Time limit: 0.1 s	Memory limit: 64 MB
-------------------	-------------------	---------------------

In this task we consider sequences of numbers $S_1, S_2, \dots$ of increasing length. The first sequence S_1 is just the number one

$$S_1 = 1$$

The following sequences S_i ($i > 1$) are, formed by two copies of the previous sequence S_{i-1} combined by the number i . Formally, the i -th sequence S_i ($i > 1$) can be recursively defined as

$$S_i = S_{i-1}, i, S_{i-1}$$

Note that, according to this definition, the first three sequences are:

$$S_1 = 1$$

$$S_2 = 1, 2, 1$$

$$S_3 = 1, 2, 1, 3, 1, 2, 1$$

$$S_4 = \dots$$

Your task is to write a program which can find the k -th number of the n -th sequence S_n . The numbers of any sequence are numbered starting from 1.

Input

The input consists of a single line containing two integers k and n , $1 \leq k, n \leq 2^{63} - 1$, separated by a single space.

Output

You must output a single line with a single integer, the k -th number of the sequence S_n . Note that it is possible, that this number is not defined (for example when k is larger than the length of the sequence S_n). In such a case you should output a single line with the string "No solution" (quotes for clarity) instead.

Partial Scoring

For testcases, which are together worth at least 30% of the points, it holds that $k \leq 2^{20}$.

Sample tests

input	output
2 2	2
input	output
4 3	3
input	output
4 2	No solution



Map Generator

Task ID: map	Time limit: 1.3 s	Memory limit: 64 MB
--------------	-------------------	---------------------

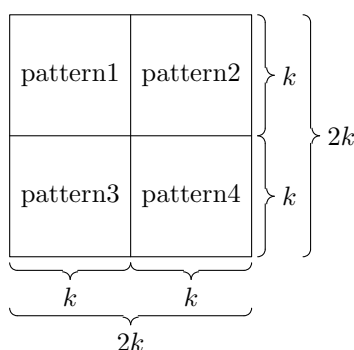
Peter is writing the random map generator for the new game *Heroes of Mouse and Keyboard*. The map for the game has the structure of a square grid of size $2^m \times 2^m$, each unit square of which may be filled with either land, or water.

Peter's generator creates the map in form of a *map description*. The map description has the following structure. Initially there are two patterns: '1' and '0', standing for a unit square of land and water, respectively. These patterns have size 1.

The map description is then a series of a pattern-definitions of the following form:

"<pattern>=<pattern1>,<pattern2>,<pattern3>,<pattern4>"

Here <pattern1> through <pattern4> are names of already declared patterns of size k (all the patterns of the right hand side of the rule are of the same size). After this rule is processed, the new pattern <pattern> (which has size $2k$) becomes defined. This pattern is constructed by putting the patterns from the right-hand side of the rule together to form a square of size $2k \times 2k$, as shown in the picture below.



The map description contains multiple such pattern definitions. The final map will be the last defined pattern (denoted by "Map").

Peter has generated his map and looks at the outcome. He is now trying to figure out, whether his new map will be fun to play or not. We define a *connected component* of land of the map as all tiles of land, which are connected either horizontally or vertically. Consider the following map of size 4×4 with two connected components of land, one of size 5 and another of size 4 (remember that tiles of land are denoted by '1').

1	1	0	1
1	1	0	1
0	1	0	1
0	0	0	1

Peter knows that the best maps for *Heroes of Mouse and Keyboard* have a very specific number of connected components. Help him to count the number of connected components of his own map.



Input

The input contains the description of the map in the format given above. Each line of the input consists of the definition of a pattern starting with the name of the new pattern followed by “=” and four patterns, separated by commas. Note that there are no spaces in between patterns. Moreover, the names of the patterns consist of letters of the english alphabet and digits only. All names are case-sensitive and not exceeding 20 characters in length. A newly defined pattern will never have the same name as an already defined one. The maximal size of any pattern is $2^{16} \times 2^{16}$ and the number of patterns does not exceed 200.

The map itself is specified by the pattern with the special name “Map” and described on the last line of the input. There are no unneeded patterns, i.e. each pattern is used (either directly or indirectly) to create “Map” (except, possibly, “0” and “1”). Every pattern will only be used after it has defined. All quotes are used for clarity only and do not appear in the input.

Output

Output a single line with a single integer, the number of connected components of land in the map.

Partial Scoring

For testcases, which are together worth at least 30% of the points, it holds that the map will be of size at most $2^{10} \times 2^{10}$.

Sample tests

input

```
A=0,1,1,0
X=1,1,1,1
B=A,A,A,X
C=B,B,B,B
Map=C,C,C,C
```

output

56

input

```
p=1,1,1,1
p24=0,1,0,1
P=0,1,0,0
Map=p,p24,P,p24
```

output

2

Remark: The second sample input corresponds to the example given above.

**Almost Palindromic Numbers**

Task ID: almost	Time limit: 0.3 s	Memory limit: 64 MB
-----------------	-------------------	---------------------

A *palindromic* number is a number, which is equivalent to its reverse. The number 9887889, for example, is palindromic. Let us call an integer *almost palindromic* if it can be converted to a palindromic number by replacing at most one digit by some other digit. For example, 1234321, 1234311, 123421 are almost palindromic, but 1234213 or 12345331 are not.

Given a number a , find the number of almost palindromic numbers smaller or equal to a . That is, find the number of values x , $1 \leq x \leq a$, which are almost palindromic.

Input

The input consists of a single line with a single integer a , $1 \leq a \leq 10^{18}$.

Output

Your program should output a single line with a single integer – the number of almost palindromic numbers smaller or equal to a .

Partial Scoring

For testcases, which are together worth at least 40% of the points, it holds that $a \leq 10^6$.

Sample tests

input	output
1000	1000
input	output
1000000	45010

**Balls and Boxes**

Task ID: boxes	Time limit: 0.05 s	Memory limit: 64 MB
----------------	--------------------	---------------------

There are n boxes standing in a circle, numbered from 1 to n in clockwise direction. Each box has a neighboring box on its *left* (box $i - 1$ if $i > 1$ and box n otherwise) and a neighboring box on its *right* (box $i + 1$ if $i < n$ and box 1 otherwise).

Initially, there are a_i balls in the i -th box, $1 \leq i \leq n$. It is known that the total number of balls $t = \sum_{i=1}^n a_i$ does not exceed n . We will consider a series of actions. In one action it is possible to move exactly one ball from a box to one of its neighboring boxes (left or right). What is the minimum number of actions needed, to distribute the balls, such that each box contains at most one ball? Note that balls can be moved multiple times, but each time it counts as one action.

Input

The first line of the input consists of an integer n , $1 \leq n \leq 1000$, the number of boxes. The second line of the input contains n integer numbers $a_1, a_2, \dots, a_n$, $0 \leq a_i \leq n$, separated by one space each. The i -th of those integers describes the number of balls a_i initially in the i -th box. For the sum $t = \sum_{i=1}^n a_i$ it holds that $0 \leq t \leq n$.

Output

The output must consist of a single line with a single integer, the minimum number of actions needed, so that no box contains more than one ball.

Partial Scoring

For testcases, which are together worth at least 50% of the points, it holds that $n \leq 100$.

Sample tests

input

7
1 0 0 0 2 3 1

output

7
