A. Goshtasp, Vishtasp and Eidi

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

Goshtasp was known to be a good programmer in his school. One day Vishtasp, Goshtasp's friend, asked him to solve this task:

Given a positive integer n, you should determine whether n is rich.

The positive integer x is rich, if there exists some set of distinct numbers $a_1, a_2, \ldots, a_m$ such that $\sum_{i=1}^{m} a_i = x$. In addition: every a_i should be either a prime number, or equal to 1.

Vishtasp said that he would share his Eidi $50 / 50$ with Goshtasp, if he could solve the task. Eidi is money given to children for Noruz by their parents and/or relatives.

Goshtasp needs to solve this problem to get money, you need to solve it to get score!

Input

Input contains a single positive integer n ($1 \le n \le 10000$).

Output

If the number is not rich print 0. Otherwise print the numbers $a_1, \ldots, a_m$. If several solutions exist print the lexicographically *latest* solution. Answers are compared as sequences of numbers, not as strings.

For comparing two sequences $a_1, \ldots, a_m$ and $b_1, \ldots, b_n$ we first find the first index i such that $a_i \neq b_i$, if $a_i < b_i$ then a is lexicographically earlier and if $b_i < a_i$ then b is lexicographically earlier. If $m \neq n$ we add zeroes at the end of the smaller sequence (only for the moment of comparison) and then perform the comparison.

You do not need to minimize the number of elements in sequence (i.e. m). You just need to print the lexicographically *latest* solution.

See samples to find out how to print the sequence.

Examples

input
```
11
```
output
```
11=11
```

input
```
545
```
output
```
541+3+1=545
```

B. INI-file

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

The INI file format is a de facto standard for configuration files. INI files are simple text files with a basic structure. They are commonly associated with Microsoft Windows, but are also used on other platforms.

Each line in INI-file stands for key-value mapping or defines new section. A key-value line has a format "`key=value`",where `key` — is the name of some property, and `value` — it's value. It is possible that it will be spaces from the both sides of `key` and/or `value`, the spaces should be ignored.

A section line has a format "`[section]`". It means that all key-value lines after it define properties of the specified section. Of cause, the following section line changes the current section. A section line may have spaces around any of brackets.

Also you should ignore comment lines — the first non-space character of comment line is "`;`".

You task is to write the program which will format given INI-file in a special way:

- first, print key-value lines which do not belong to any section;
- print all the sections in the lexicographical (alphabetical) order of their names;
- inside each of two previous items, order key-value lines lexicographically by "`key`";
- if there are more than one key-value lines with the same key inside a single section (or outside any sections), leave only one line (which appears later in the input data);
- remove all redundant spaces and lines.

Input

The first line contains single integer n ($1 \le n \le 510$) — the number of lines in given INI-file.

The rest of the input contains a valid INI-file in n lines. Values of `section`, `key` and `value` contain only Latin letters, digits, "`.`" and/or "`-`".

Each line has length not exceeding 255 characters and not less than 1 character. The total length of all the lines does’t exceed 10000.

Output

Print formatted INI-file.

Examples

input

```
11
a= 1
b=a
a = 2
 ; comment
[z]
1=2
[y]
2=3
[z]
2=1
[w]
```

output

```
a=2
b=a
[w]
[y]
2=3
[z]
1=2
2=1
```

C. Extraordinarily Nice Numbers

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

The positive integer a is a divisor of the positive integer b if and only if there exists a positive integer c such that $a \times c = b$.

King Astyages thinks a positive integer x is *extraordinarily nice* if the number of its even divisors is equal to the number of its odd divisors.

For example 3 has two positive divisors 3 and 1, both of which are odd, so 3 is not extraordinarily nice. On the other hand 2 is only divisible by 2 and 1, so it has one odd and one even divisor. Therefore 2 is extraordinarily nice.

Given a positive integer x determine whether it's extraordinarily nice.

Input

The input contains only a single integer x ($1 \leq x \leq 10^3$).

Output

Write a single yes or no. Write yes if the number is extraordinarily nice and no otherwise.

You don't need to care about capital or small letters. The output is case-insensitive.

Examples

input

```
2
```

output

```
yes
```

input

```
3
```

output

```
no
```

D. Perse-script

time limit per test: 7 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Two good friends were trying to make a new programming language called Perse-script.

The most important part of this language is *strings*. A string in Perse-script is put between characters "

So for example `"Hello"` is a string. But `Hello` is a variable name or a keyword, which are not considered in this problem.

Perse-script is function-based. So there are no operators in this language. For example for summing two numbers you have to write `sum(a,b)` and not `a+b`.

There are several functions for working on strings. These include:

- `concat(x,y)` is a function which gets two strings x and y and puts y at the end of x and returns the result. For example `concat("Hello","World")` returns `"HelloWorld"`.
- `reverse(x)` gets a single string x and reverses it. For example `reverse("Hello")` returns `"olleH"`.
- `substr(x,a,b)` gets a string x and two integers a and b ($1 \le a \le b \le n$, where n is the length of x). And returns the substring of x between indexes a and b, inclusive. For example `substr("Hello",2,4)` returns `"ell"`.
- `substr(x,a,b,c)` is another version of substr which works just like the last one but c is the step of adding. c is *positive*. For example `substr("HelloWorld",1,10,2)` returns `"Hlool"`. This substr means that you put the ath character , and then every cth character until you reach b.

You're going to manipulate the string part of Perse-script. Given a string expression, you should print its result. It is guaranteed that the expression contains only strings shown by characters " and the above functions.

Commands in Perse-script are case-insensitive. So to call `substr` function you can write `SUBsTr()`. But you can't print as the result `"hElLo"` instead of printing `"Hello"`.

See the samples for more information.

Input

A single line containing the correct expression. It is guaranteed that the total length of this expression does not exceed 10^3 and that all the integers used in it are less than or equal to 100 by absolute value. The given string is non-empty.

All strings in the input which are placed between "s consist of uppercase and lowercase Latin letters only.

Output

Print in single line the resulting string. It is guaranteed that an answer exists and that the length of the answer does not exceed 10^4. It is guaranteed that the answer is non-empty.

Examples

input

```
"HelloWorld"
```

output

```
"HelloWorld"
```

input

```
REVerse(substr("helloworld",1,5))
```

output

```
"olleh"
```

input

```
conCAT(rEveRSE("olleh"),"world")
```

output

```
"helloworld"
```

input

```
reversE(concAT(substr("hello",1,2),sUbstr("world",1,5,1)))
```

output

```
"dlroweh"
```

input

```
suBstr("Noruz",1,4,2)
```

output

```
"Nr"
```

E. Ali goes shopping

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

Ali Koochooloo is going to buy new clothes since we're reaching Noruz, the ancient Persian festival and the beginning of new Persian year.

When Ali entered a shop, he saw that the shopkeeper was a programmer and since there is no money in programming he had changed his career. The shopkeeper told Ali that he can buy anything for free if he could answer a simple question in 10 seconds. But to see the question Ali has to pay 3 tomans.

Ali agreed instantly and the shopkeeper handed him a piece of paper containing the task. The task was indeed very simple. It said:

Let string A be ababababababab. Which non-empty substring of A is repeated the most times in it?

Ali answered fast. He said the answer is a. But the shopkeeper said that Ali is wrong and asked him to read the rest of statement:

If several substrings have the maximal repeat time, then the substring with maximal length would be the answer, in case of a tie the alphabetically latest substring will be chosen.

So the answer is ab.

Now Ali wants us to solve this problem for different strings. We don't have a great advantage over Ali, we just have a computer and a weird language.

Input

The single line consisting of a string A. It is non-empty, made of lower-case Latin letters and contains at most 30 characters.

Output

The single line contains the answer.

Examples

input

```
abab
```

output

```
ab
```

input

```
abcd
```

output

```
abcd
```

F. Oil

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

After the nationalization of the oil industry, Dr. Mosaddegh wants to dig some oil wells to extract all the oil in Persian Gulf. But Persian Gulf is huge and has an infinite amount of oil. So Dr. Mosaddegh works only on a rectangular plane of size $n \times m$ of the Persian Gulf. Each of the cells in this rectangle either contains an infinite amount of oil or nothing.

Two cells are considered adjacent if and only if they have a common edge, a path is a sequence $c_1, c_2, \dots, c_x$ of cells so that all of them contain oil and for each i, c_i is adjacent to c_{i-1} and c_{i+1} (if they exist). Two cells are considered *connected* to each other if and only if there exists a path between them. If we dig a well in a certain cell, we can extract oil from all the cells that are *connected* to it by oil paths. It is not allowed to dig wells on empty cells.

Dr. Mosaddegh also knows that in Persian Gulf, the empty cells form rows and columns. I. e. if some cell is empty, then it's column is completely empty or it's row is completely empty, or both.

Help Dr. Mosaddegh find out how many wells he has to dig to access all the oil in that region.

Input

In the first line there are two positive integers n and m ($1 \le n, m \le 100$).

In the second line there is an integer t ($0 \le t \le n$), the number of empty rows. t distinct positive integers follow, these are the numbers of empty rows and are in range $[1, n]$.

In the second line there is an integer s ($0 \le s \le m$) that shows the number of columns not having any oil. s distinct positive integers follow, these are the numbers of empty columns and are in range of $[1, m]$.

Note that rows are numbered from 1 to n (from top to bottom) and columns are numbered from 1 to m (from left to right).

Output

A single integer, the minimum number of wells that Dr. Mossadegh has to dig.

This is actually finding how many regions are made by removing the given rows and columns.

Examples

input

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

output

```
2
```

input

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

output

```
2
```

input

```
2 3
1 1
0
```

output

```
1
```

G. Fibonacci army

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

King Cambyses loves Fibonacci numbers. He has several armies. Today he wants to make a new army for himself and he wants the number of men in this army to be the n-th Fibonacci number.

Given n you should find n-th Fibonacci number. The set of Fibonacci numbers start with $f_0 = f_1 = 1$ and for each $i \geq 2$, $f_i = f_{i-1} + f_{i-2}$.

Input

Input contains a single integer n ($1 \leq n \leq 20$).

Output

Write a single integer. The n-th Fibonacci number.

Examples

input

```
2
```

output

```
2
```

input

```
1
```

output

```
1
```

H. Reverse It!

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

The 14th of March was the international day of mathematics, because of number $\pi = 3.1415926...$.

In the occasion of this day Goofy Nephews Unity Organization (GNU) wants to publish the fastest program in math at 1:59:26 AM.

Now the time is 1:11:11 AM and the project team haven't checked their program yet. Because of shortage of time they want to check their program with some queries. So they hired Hormizd (one of the greatest programmers in world) to write a tester for GNU's new program. Because Hormizd has much more important things to do, he wants you to write a small part of tester and it is reversing the numbers. Help him before 1:59:26.

We can reverse numbers easily. For example by reversing 1234 we get 4321.

Note, that if the integer is negative then its reverse would be also negative. For example reverse of -123 is -321.

Also, you have to delete all the leading zeroes before and after the reverse.

Given an integer you have to help Hormizd reverse it.

Input

The first line contains a single integer n. It is less than 10^{1000} by it's absolute value. This integer may have leading zeros. If it had leading zeros you should first omit them and print the reverse of remaining digits. It's guaranteed that input contains less than 10001 characters.

Output

Output a single integer, the reverse of the given number. You have to omit leading zeros in the output.

Examples

input
```
23
```
output
```
32
```

input
```
-032
```
output
```
-23
```

input
```
01234560
```
output
```
654321
```

I. Goofy Numbers

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

The non-negative integer a is a divisor of the non-negative integer b if and only if there exists a **positive** integer c such that $a \times c = b$.

Some numbers are really interesting. Commander Surena defines some interesting properties for non-negative integers:

- An integer is *happy* if it is divisible by at least one of its digits and not by all of them.
- An integer is *happier* if it is divisible by all of its digits.
- An integer is *upset* if it's divisible by none of its digits.

Surena asks you to find out if a given number is happy, happier or upset.

Input

Input contains a single non-negative integer n ($1 \le n \le 10^8$).

Output

Write on a single line the type of the integer: `happy`, `happier` or `upset`. Print the type in lowercase letters.

Examples

input
```
99
```
output
```
happier
```

input
```
29994
```
output
```
happy
```

input
```
23
```
output
```
upset
```

Note

In the second test 29994 is only divisible by 2.

In the third test 23 is a prime number.