A. Pentagonal numbers

time limit per test: 3 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

Pentagonal numbers are figurate numbers which can be calculated using the formula $p_n = (3n^2 - n) / 2$ (always integer). You are given n; calculate n-th pentagonal number.

Input

The only line of input contains an integer n ($1 \le n \le 100$).

Output

Output the n-th pentagonal number.

Sample test(s)

input

```
2
```

output

```
5
```

input

```
5
```

output

```
35
```

B. Binary notation

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

You are given a positive integer n. Output its binary notation.

Input

The only line of input data contains an integer n ($1 \leq n \leq 10^6$).

Output

Output the binary notation of n (without any leading zeros).

Sample test(s)

input

```
5
```

output

```
101
```

input

```
126
```

output

```
1111110
```

Note

In the first example $5 = 1 * 2^2 + 0 * 2^1 + 1 * 2^0$.

C. Prime factorization

time limit per test: 3 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

You are given a positive integer n. Output its prime factorization.

If $n = a_1^{b_1} a_2^{b_2} \ldots a_k^{b_k}$ $(b_i > 0)$, where a_k are prime numbers, the output of your program should look as follows: $a_1{*}\ldots{*}a_1{*}a_2{*}\ldots{*}a_2{*}\ldots{*}a_k{*}\ldots{*}a_k$, where the factors are ordered in non-decreasing order, and each factor a_i is printed b_i times.

Input

The only line of input contains an integer n $(2 \le n \le 10000)$.

Output

Output the prime factorization of n, as described above.

Sample test(s)

input

```
245
```

output

```
5*7*7
```

input

```
19
```

output

```
19
```

D. Remove digits

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

You are given a string. Remove all digits from it. When a character is removed from a string, all characters to the right of it are shifted one position to the left.

Input

The only line of input contains a string between 1 and 100 characters long. Each character of the string has ASCII-code between 33 (exclamation mark) and 126 (tilde), inclusive.

Output

Output the given string with all digits removed from it. If the original string had only digits, output an empty string.

Sample test(s)

input

```
VK-Cup-2012!
```

output

```
VK-Cup-!
```

input

```
Go,Codeforces!
```

output

```
Go,Codeforces!
```

E. HQ9+

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

HQ9+ is a joke programming language which has only four one-character instructions:

- "`H`" prints "`Hello, World!`",
- "`Q`" prints the whole source code of the program itself (at each call),
- "`9`" prints the lyrics of "99 Bottles of Beer" song,
- "`+`" increments the value stored in the internal accumulator.

Instructions "`H`" and "`Q`" are case-sensitive and must be uppercase. The characters of the program which are not instructions are ignored.

You are given a program written in HQ9+. You have to figure out whether executing this program will produce any output.

Input

The input will consist of a single line p which will give a program in HQ9+. String p will contain between 1 and 100 characters, inclusive. ASCII-code of each character of p will be between 33 (exclamation mark) and 126 (tilde), inclusive.

Output

Output "`YES`", if executing the program will produce any output, and "`NO`" otherwise (quotes for clarity only).

Sample test(s)

input

```
Hello!
```

output

```
YES
```

input

```
VK_Cup_2012!
```

output

```
NO
```

Note

In the first case the program contains only one instruction — "`H`", which prints "`Hello, World!`".

In the second case none of the program characters are language instructions.

F. Factorial zeros

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

You are given a positive integer n. Output the number of trailing zeros in $n!$ ($n!$ denotes a product of integers between 1 and n, inclusive).

Input

The only line of input contains an integer n ($1 \le n \le 1000000$).

Output

Output the number of trailing zeros in $n!$.

Sample test(s)

input

```
6
```

output

```
1
```

input

```
24
```

output

```
4
```

Note

In the first sample $6! = 720$.

In the second sample $24! = 620448401733239439360000$.

G. Non-decimal sum

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

You are given an array of integers written in base $radix$. Calculate their sum and output it written in the same base.

Input

The first line of the input contains an integer n $(1 \leq n \leq 10)$ — the size of the array. The second line contains an integer $radix$ $(2 \leq radix \leq 36)$ — the base of the numeral system used. Next n lines contain the elements of the array, one per line.

Each element is a non-negative integer written in $radix$-based notation, possibly with leading zeros, which contains between 1 and 5 digits, inclusive. The digits of the notation will be 0, 1, ..., 9, A, B, ..., Z in the given order.

Output

Output the sum of array elements in $radix$-based notation. Use the same format as in the input.

Sample test(s)

input

```
3
16
F0
20B
004
```

output

```
2FF
```

input

```
2
10
12
34
```

output

```
46
```

H. Alternating case

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

You are given a string consisting of alphabet letters. Convert it to alternating case: the letters on odd positions should be in uppercase, and the letters on even positions should be lowercase. The letters are numbered staring from 1.

Input

The only line of input contains a string between 1 and 100 characters long. Each character of the string is either an uppercase ('A'-'Z') or a lowercase ('a'-'z') letter.

Output

Output the resulting string.

Sample test(s)

input

```
Codeforces
```

output

```
CoDeFoRcEs
```

input

```
VKCup
```

output

```
VkCuP
```

I. Truncatable primes

time limit per test: 3 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

A truncatable prime is a prime number which contains no zeros in decimal notation and all its suffixes are primes. 1 is considered to be not a prime.

You are given a positive integer n. Figure out whether it is a truncatable prime.

Input

The only line of input contains an integer n ($2 \leq n \leq 10^7$).

Output

Output "YES" if n is a truncatable prime. Output "NO" otherwise. Quotes for clarity only.

Sample test(s)

input

```
19
```

output

```
NO
```

input

```
9137
```

output

```
YES
```

Note

In the first sample 19 is a prime but its suffix 9 is not.

In the second sample 9137, 137, 37 and 7 are all primes, so 9137 is a truncatable prime.

J. Brackets

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

A sequence of brackets is called balanced if one can turn it into a valid math expression by adding characters "+" and "1". For example, sequences "(())()", "()" and "(()(()))" are balanced, while ")(", "(()" and "(()))(" are not.

You are given a string which consists of opening and closing round brackets. Check whether it is a balanced bracket sequence.

Input

The only line of input contains a string between 1 and 100 characters long, inclusive. Each character in the string will be "(" or ")".

Output

Output "YES" if the bracket sequence is balanced, and "NO" otherwise (quotes for clarity only).

Sample test(s)

input

```
(()(()))()
```

output

```
YES
```

input

```
())()
```

output

```
NO
```