April Fools Contest 2018

A. Quirky Quantifiers

time limit per test: 2 seconds
memory limit per test: 64 megabytes
input: standard input
output: standard output

Input
The input contains a single integer a ($10 \le a \le 999$).

Output
Output 0 or 1.

Examples

input
13
output
1

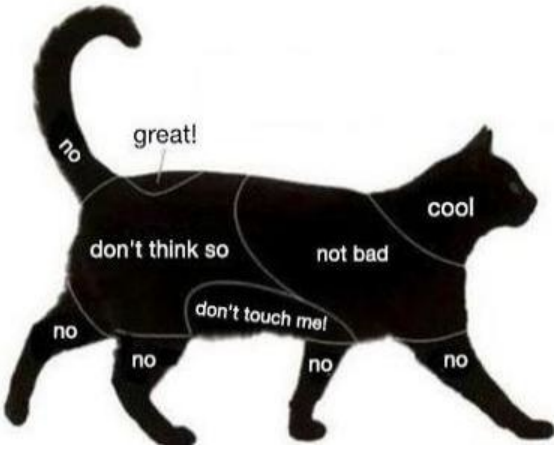
input
927
output
1

input
48
output
0

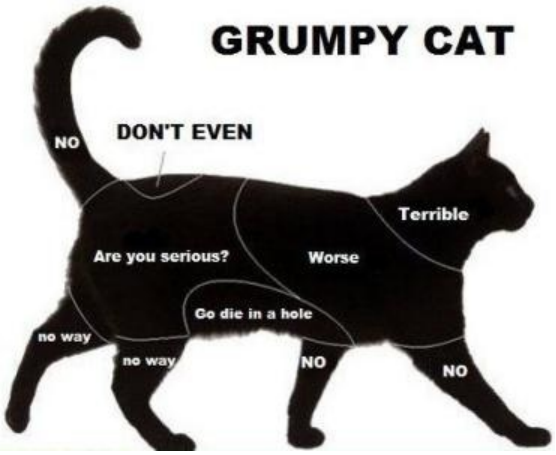
B. A Map of the Cat

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

If you have ever interacted with a cat, you have probably noticed that they are quite particular about how to pet them. Here is an approximate map of a normal cat.



However, some cats won't tolerate this nonsense from the humans. Here is a map of a grumpy cat.



You have met a cat. Can you figure out whether it's normal or grumpy?

Interaction

This is an interactive problem. Initially you're not given any information about the cat. Instead, the cat is divided into ten areas, indexed from 0 to 9.

In one query you can choose which area you'll pet and print the corresponding index to standard out. You will get the cat's response, as depicted on the corresponding map, via standard in. For simplicity all responses are written in lowercase.

Once you're certain what type of cat you're dealing with, output "normal" or "grumpy" to standard out.

Note

Please make sure to use the stream flushing operation after each query in order not to leave part of your output in some buffer.

C. Ravioli Sort

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Everybody knows of spaghetti sort. You decided to implement an analog sorting algorithm yourself, but as you survey your pantry you realize you're out of spaghetti! The only type of pasta you have is ravioli, but you are not going to let this stop you...

You come up with the following algorithm. For each number in the array a_i, build a stack of a_i ravioli. The image shows the stack for $a_i = 4$.

Arrange the stacks in one row in the order in which the corresponding numbers appear in the input array. Find the tallest one (if there are several stacks of maximal height, use the leftmost one). Remove it and add its height to the end of the output array. Shift the stacks in the row so that there is no gap between them. Repeat the procedure until all stacks have been removed.

At first you are very happy with your algorithm, but as you try it on more inputs you realize that it doesn't always produce the right sorted array. Turns out when two stacks of ravioli are next to each other (at any step of the process) and differ in height by two or more, the top ravioli of the taller stack slides down on top of the lower stack.

Given an input array, figure out whether the described algorithm will sort it correctly.

Input

The first line of input contains a single number n ($1 \le n \le 10$) — the size of the array.

The second line of input contains n space-separated integers a_i ($1 \le a_i \le 100$) — the elements of the array.

Output

Output "YES" if the array can be sorted using the described procedure and "NO" if it can not.

Examples

input
3 1 2 3
output
YES

input
3 3 1 2
output
NO

Note

In the second example the array will change even before the tallest stack is chosen for the first time: ravioli from stack of height 3 will slide on the stack of height 1, and the algorithm will output an array $\{2, 2, 2\}$.

D. I'm Feeling Lucky!

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have one chip and one chance to play roulette. Are you feeling lucky?

Output

Print your bet. Your chip must be placed entirely within some square (not on an edge or a corner shared by adjacent squares).

E. Cheese Board

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Not to be confused with chessboard.

Input
The first line of input contains a single integer N ($1 \le N \le 100$) — the number of cheeses you have.

The next N lines describe the cheeses you have. Each line contains two space-separated strings: the name of the cheese and its type. The name is a string of lowercase English letters between 1 and 10 characters long. The type is either "`soft`" or "`hard`". All cheese names are distinct.

Output
Output a single number.

Examples

input
9 brie soft camembert soft feta soft goat soft muenster soft asiago hard cheddar hard gouda hard swiss hard
output
3

input
6 parmesan hard emmental hard edam hard colby hard gruyere hard asiago hard
output
4

F. 2 + 2 != 4

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

One very experienced problem writer decided to prepare a problem for April Fools Day contest. The task was very simple - given an arithmetic expression, return the result of evaluating this expression. However, looks like there is a bug in the reference solution...

Input
The only line of input data contains the arithmetic expression. The expression will contain between 2 and 10 operands, separated with arithmetic signs plus and/or minus. Each operand will be an integer between 0 and 255, inclusive.

Output
Reproduce the output of the reference solution, including the bug.

Examples

input
8-7+6-5+4-3+2-1-0
output
4

input
2+2
output
-46

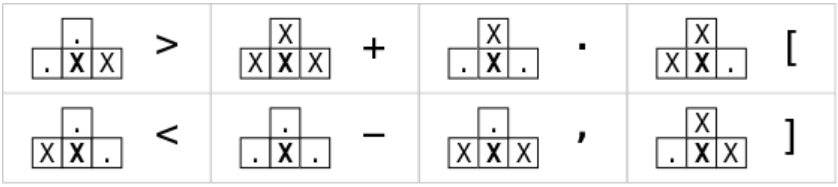
input
112-37
output
375

G. Puzzling Language

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

In this problem you will write a simple code generator for a 2D programming language derived from Brainfuck.

The code in this language is a rectangular grid of characters '.' and 'X'. The code is converted to a Brainfuck program as follows: the characters are read in the usual order (top to bottom, left to right), and each 'X' character is converted a Brainfuck instruction to be executed. The instruction is defined by the left, top and right neighbors of the 'X' character using the following conversion table:



You are given a string. Output a program in the described language which prints this string.

You can download the language interpreter used for judging here: https://assets.codeforces.com/rounds/952/puzzling-interpreter.cpp (use C++11 to compile the code). Note several implementation details:

- The first step of the language interpretation is conversion to a Brainfuck program, which is then executed.
- The code must be rectangular, with all lines of the same length. It can have at most 10,000 lines and 10,000 columns, and can have at most 500,000 'X' characters.
- The code has toroidal topology, i.e. the 'X' on the first line will have top neighbor in the last line.
- Brainfuck interpreter has 30000 memory cells which store integers from 0 to 255 with increment/decrement done modulo 256.
- Console input (`,` command) is allowed in Brainfuck code but has no effect when executed.

Input
The input consists of a single string of characters with ASCII codes between 33 ('!') and 122 ('z'), inclusive. The length of the string is between 1 and 10 characters, inclusive.

Output
Output a program in the described language which, when executed, will print the given message.

Example

input
$$$
output
.......X....... ......XXX...... .....XXXXX..... ....XXXXXXX.... ...XXXXXXXXX... ..XXXXXXXXXXX.. .XXXXXXXXXXXXX. .... X...........X X............. X............. X.............</br>

Note
The example corresponds to the following Brainfuck program:

```
       -
      >+<
     >+++<
    >++++++<
   >++++++++<
  >++++++++++<
 >++++++++++++<

<             >
.
.
.
```

The triangular block decrements the first memory cell and sets the value of the second memory cell to 36 - the ASCII code of '$' character. The next line after the triangular block moves the memory pointer to the second memory cell, and the next three lines print the '$' character three times.