Problem D. The Diputs notation

Input file:	`diputs.in`
Output file:	`diputs.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

The Diputs notation is a notation to represent non-negative integer numbers. Diputs notation has 9 digits, starting with the least significant:

```
_.,-~=’^"
```

There can be a limited quantity of each digit character in the number representation:

Order of digit	Character	ASCII code	Max. quantity
1	`_`	95	2
2	`.`	46	3
3	`,`	44	5
4	`-`	45	7
5	`~`	126	11
6	`=`	61	13
7	`’`	39	17
8	`^`	94	19
9	`"`	34	23

A number representation in Diputs notation starts with the most significant digit, e.g.:

```
"""^^~~-,..__
```

Let us write down all possible number representations in Diputs notation and arrange them in the following order. Number A goes after number B if the representation of A in Diputs notation has more ‘`"`’ (most significant digit) characters than the representation of B. If the numbers A and B have equal number of ‘`"`’ characters, we then compare the number of ‘`^`’ characters, and so on. Then we number this sequence, starting from 1 — and the result is the map which defines how numbers are represented in Diputs notation. Zero is represented with the ‘`O`’ character (code 79).

Examples:

Decimal notation	Diputs notation
0	`O`
1	`_`
2	`__`
3	`.`
4	`._`
6	`..`
20	`,..__`
34836480	`"`
36682648	`"^’=~-,._`

There is an input file containing numbers in decimal notation, numbers in Diputs notation, and some other text. You must output the file containing the same numbers, but in another notation: the numbers in decimal notation should be converted to Diputs notation, and the numbers in Diputs notation should be converted to decimals. All numbers in the input file are non-negative and do not exceed maximum number representable in Diputs notation. It is possible that the numbers will not be separated by space or other characters. You should parse the numbers in greedy way: when reading a number, you should take maximum number of characters that can form a decimal or Diputs representation of a number. For example, if you have three underscore characters (“`___`”), you should parse it as numbers “`__`” (2) and “`_`” (1) in Diputs notation. In this case, the resulting string will be “`21`”. When parsing decimal numbers, your program should work the same way. There should be no leading zeros in decimal numbers. For example, you should parse the string “`020`” as decimal numbers 0 and 20, and the result will be “`0,..__`”. If you would get a decimal number larger than the maximum number representable in Diputs notation, you should split it into two or more decimal numbers in greedy way, as described.

Some examples:

Input	Output
`___`	`21`
`020`	`0,..__`
`_12`	`1,`

Because we feel that your task is too easy, you should sort the numbers in ascending order, if the total number of numbers in the input file is odd. Only numbers should be reordered; the notation of a number in a certain location should not be changed. For example, for input “`2 0 _`” you should first recognize the notation (decimal, diputs, diputs), then convert it to “`__ 0 1`”, inverting the notation (it is now diputs, decimal, decimal), and then sort the numbers while preserving the notation. Thus, the sorted array is 0, 1 and 2, the notations are (diputs, decimal, decimal), so the answer should be “`0 1 2`”. Take a look at the sample input for better understanding.

Size of input and output files will not exceed 1 megabyte (10^6 bytes).

Examples

diputs.in	diputs.out
`need 2 sort`	`need 0 sort`
`0`	`1`
`_`	`2`
`1`	`_`
`2`	`__`
`020`	`0,..__`
`___`	`21`
`-A-`	`72A72`
`,._`	`16`
`0`	`0`