As we know, in an n-based number system, there are n different types of digits. In this way, a 1-based number system has only 1 type of digit, the '0'. Here are the rules to interpret 1-based numbers. Each number consists of some space separated blocks of 0. A block may have 1, 2 or more 0's. There is a 'flag' variable associated with each number

- A block with a single '0' sets 'flag' variable to 1
- A block with two 0's sets the 'flag' to 0
- If there are n ($n > 2$) 0's in a block, $n-2$ binary digits with the current value of flag is appended to your number.

Note that, the first block of every number will have at most 2 0s. For example, the 1-base number '0 0000 00 000 0 0000' is equivalent to binary '11011'.

- 1st block sets the flag to 1
- 2nd block has 4 0's. So append flag(=1) 4 2 = 2 times (11).
- 3rd block has 2 0's. Set the flag to 0
- 4th block has 3 0's. Append flag(=0) 3-2 = 1 time (110).
- 5th block has a single '0'. Set flag = 1
- 6th and block has 4 0's. Append flag(=0) 4-2=2 times (11011).

The final binary number wont have more than 30 digits. Once, youve completed the process, convert the binary value to decimal and print, youre done!

Input

Input will have at most 100 test cases. Each case consists of a 1-based number as described above. A number may be spanned to multiple lines but a single block will always be in a single line. Termination of a case will be indicated by a single '#' char which will be space-separated from the last digit of your input number. The last case in the input is followed by a '~' character indicating, end of input.

Output

For each test case, output a single line with the decimal equivalent value of your given 1-based number.

Sample Input

```
0 0000 00 000 0 0000 #
0 000 #
~
```

Sample Output

```
27
1
```