This problem doesn't have a story. You just have to write a calculator which could perform simple binary operations. You will be given an expression and your program should calculate it's result. Expression is formed according to this rules:

```
<digit> ::= 0|1
<number> ::= <digit>|<digit><number>
<unary_operator> :== not | shr | shl
<binary_operator> :== xor | and | or
<token> ::= <number> | <unary_operator> <token> | <token> <binary_operator> <token>
<expression> ::= <token> | <token> <expression>
```

Operation definitions:
`not` – binary negation operation (example: `not 101 = 010`)
`shr` – binary right shift operation (example: `shr 101 = 10`)
`shl` – binary left shift operation (example: `shl 101 = 1010`)
`xor` – binary exclusive or operation (example: `0111 xor 1011 = 1100`)
`and` – binary and operation (example: `0111 and 1011 = 0011`)
`or` – binary or operation (example: `0111 or 1011 = 1111`)

All unary operators have higher priority than binary ones and binary operators must be calculated from left to right (their priority is considered equal). Before any unary operation all additional leading zeros should be removed (for example `0011` becomes `11` and `000` becomes `0`), and before a binary one you should align binary numbers by adding leading zeros if necessary (for example `11 xor 101` becomes `011 xor 101`).

Input

The number of tests T ($T \leq 100$) is given on the first line. Each of next T lines contains an expression itself. Length of the expression will be always less than 1000 characters.

Output

For each test case output a single line '`Case` T`:` N'. Where T is the test case number (starting from 1) and N is the value of evaluated expression in the same binary form. Answer should not contain any leading zeroes.

Sample Input

```
4
shl not 101
not 11 and 111
111 xor not 0
shl 0
```

Sample Output

```
Case 1: 100
Case 2: 0
Case 3: 110
Case 4: 0
```