Binary numbers

The goal of this problem is to illusrate that all arithmetic operations on natural numbers in binary format can be implemented by using a few logical operations on the individual bits. The allowed operations are XOR, NOT, AND, OR, and SHIFT, where the latter is the operation of shifting the bits of a number to the right or to the left (with digits disappearing or new zeroes appearing at one or the other end as needed). Note that in principle, NOT, AND, and OR are superfluous because they can be expressed in terms of XOR only. No automatic check will be performed on if the submitted code satisfies the aforementioned restrictions. This means that the problem solver needs to impose the restrictions on their code, keeping in mind the goal that we are trying to mimick how the CPU hardware handles integers.

Input

The first number is the number of test cases, in binary. Then the next numbers are in groups of 3, all in binary. In each group, the first number encodes the operatioin to be performed on the next 2 numbers. The encoding convention is as follows.

0	greater than
1	addition
10	subtraction
11	multiplication
100	division

All numbers in the input are understood to be nonnegative integers. If needed, you can assume that the number of bits in each integer does not exceed 200.

Output

For each test case, the output must include one or two binary numbers. In case of division only, the output consists of two numbers: Supposing that the pair of numbers in the input are A and B, the output should be A/B (integer division) and A%B (remainder). If B happens to be 0, the output should be 0 0 (two zeroes). For the operation "greater than", output 1 if the first number is greater than the second number, and output 0 otherwise. If the first argument A of a subtraction is smaller than the second argument B, then A should be replaced by $A + 2^N$, where N is the smallest integer such that $2^N > \max\{A,B\}$. For example, in the operation 100 - 101 (dec. 4 - 5), the first argument should be increased by 1000 (dec. 8), making it 1100 (dec. 12), and the output corresponding to the operation 100 - 101 should be 111 (dec. 12 - 5 = 7).

Example

Input:

```
111
0 10 10
1 10 11
0 11 10
11 101 11
10 11 10
100 1011 11
```

```
10 10 11
```

Output:

```
0
101
1
1111
1
11 10
11
```