CSEC-ASTU Competitive Programming Contest 2021

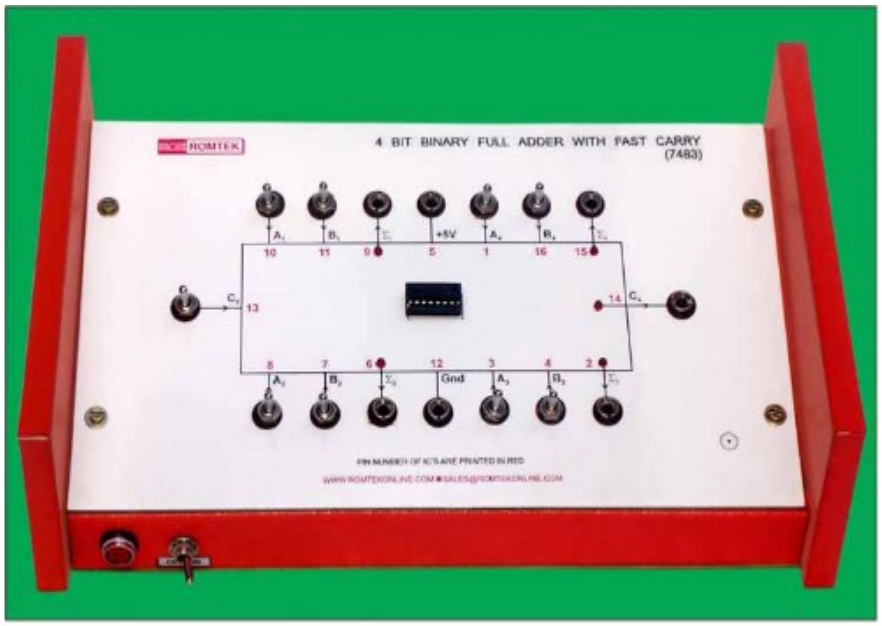
Problem N04: Parallel Carry Adder

Time limit: 1s

Let a and b be two 31-bit binary numbers. Consider the following method for computing a+b. Let c, d be 31-bit numbers such that:

- $c_i = 1$ if $a_i \neq b_i$
- $c_i = 0$ if $a_i = b_i$
- $d_i = 1$ if $a_i = b_i = 1$
- $d_i = 0$ if $a_i = 0$ or $b_i = 0$

Now, replace a with c and b with 2∗d and repeat these steps until either b= 0 or $b \leq 2^{31}$. In the former case, the resulting value of a is the sum of the original numbers. In the latter case, the sum of the original numbers requires 32 bits to express so we say that an overflow has occurred.



Your task is to print the results of all intermediate calculations and report if an overflow occurs.

Input

The first number indicates the number of test cases. Each test case consists of two 31-bit numbers a and b.

Output

The output for each test case is a series of lines. The first line consists of a and b written in binary and separated by a space. Following this, each line consists of the resulting values of a and b when one iteration of the algorithm is applied to the values in the previous line. Again, these should be written in binary and separated by a space. If any of these numbers has value at least 2^{31} then the message '`overflow`' should be printed in place of the number. The last line of output to be printed corresponds to the iteration when either b= 0 or $b \leq 2^{31}$. The output for consecutive input cases should be separated by a blank line.

CSEC-ASTU Competitive Programming Contest 2021

Sample Input 1

```
2
00000000000010000000101011101011 10000101100001100000000111111111
11000000000000000000000000000000 01000000000000000000000000000000
```

Sample output 1

```
00000000000010000000101011101011 10000101100001100000000111111111
10000101100101100000101100010100 00000000000000000000000111010110
10000101100101100000101011000010 00000000000000000000001000101000
10000101100101100000100011101010 00000000000000000000010000000000
10000101100101100000110011101010 00000000000000000000000000000000
11000000000000000000000000000000 01000000000000000000000000000000
10000000000000000000000000000000 10000000000000000000000000000000
00000000000000000000000000000000 overflow
```