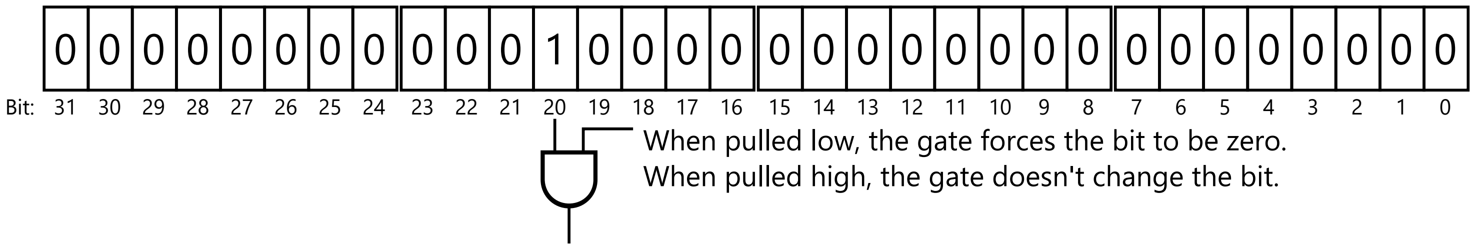
A20 Gate

Time limit: 2.0s **Memory limit:** 16M

In 1984, IBM released the PC/AT, making use of the then-new Intel 80286. The 80286 supported a whopping 16 MB of RAM, up from the old limit of 1 MB. This posed a problem, however, as older programs depended on the lack of support for any memory address over 1 MB. Since memory and processing power was scarce back then, programmers would often pull tricks to decrease code size and execution time. These programmers abused the fact that memory wrapped around after 1 MB, so accessing the memory at 1 MB would effectively access the memory at 0 instead.

IBM's solution? They added a logic AND gate on the A20 (address line 20) on the CPU. A20 transmits bit 20 of the memory address, with bit 0 being the last bit of the address. The following diagram shows the gate in action for the address at 1 MB.


0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit: 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
When pulled low, the gate forces the bit to be zero.
When pulled high, the gate doesn't change the bit.

The AND gate would make sure that A20 is *always zero* when the gate is pulled low, and have *no effect* when it's pulled high. For example, when pulled low, it makes address 1 MB behave as address 0 and address 1 MB + 1 as address 1, etc. It is important to realize that IBM did not put gates on any other address line, so that the address 2 MB does not wrap to 0, though 3 MB wraps to 2 MB.

In 1992, your second job is to find the effect the A20 gate has on a memory address. In other words, for every given memory address, output the real address accessed if the A20 gate is pulled up and down.

Input Specification

The first line of the input is the integer N such that $1 \le N \le 100$, the number of addresses you are processing. The next N lines are zero-padded 32-bit memory addresses in hexadecimal which you are to process.

Output Specification

For all the addresses that have two possible interpretations (some addresses may be the same regardless of whether the A20 is high or low), output the address in zero-padded hexadecimal when the gate is pulled low, followed by a space, followed by the address in the same format when the gate is pulled high. If the address has only one possible interpretation, output the same address back out in the same format.

Sample Input

```
3
0010DEAD
0020BEEF
0030CAFE
```

Sample Output

```
0000DEAD 0010DEAD
0020BEEF
0020CAFE 0030CAFE
```