Equivalent Addresses

Time limit: 1.0s **Memory limit:** 64M

Recall that in real mode on the x86, the programmer has access to 1 MB of memory. However, being a 16-bit mode, 16-bit pointers only allows access to 64 KB (= 2^{16} bytes) of address space. This is of course no good for accessing the 1 MB of memory, so segments were introduced. A segment is another 16-bit of data to determine where the 16-bit pointer is relative to in memory. But we only have 20-bit of data (1 MB = 2^{20} bytes, requiring 20 bits to address all memory), and we have 32-bits of data in a `segment:offset` pair, so 12-bits are redundant. **In other words, there are 2^{12} different segments, with different offsets, that give the same memory address, hence there are $2^{12} = 4\,096$ different ways to represent the same memory address.** Any given address is accessible in $4\,096$ different segments. Since switching segments is a very expensive operation, programmers are encouraged to choose a segment that has all their data visible.

The segment is multiplied by 16 before being added with the offset to yield the final address. For example, the address `06EF:1234`: it has a segment of `0x06EF`, which, multiplied by 16, gives the segment base address `0x06EF0`, and adding the offset `0x1234` to the base gives the address `0x08124`. Also recall that the lack of address lines A20 and beyond on the 8086 allows for addresses to wrap around. So segment `0xFFFF` with the offset `0x0010` becomes `0xFFFF0` + `0x0010` = `0x100000`, but wraps around to `0x00000`. This was the primary reason for the existence of the A20 gate, since older programs before A20 relied on this trick.

To help other programmers in your company to choose a nice segment for some performance boost, you received your third assignment: given a pointer in `segment:offset` format, output all the equivalent `segment:offset` pairs that refer to the same memory address, with A20 pulled low.

Input Specification

The input consists of one line, in the format `SSSS:OOOO`, where both `S` and `O` represent one hexadecimal digit. `SSSS` represents the segment and `OOOO` represents the offset of the address you are to use.

Output Specification

You are to output all the `segment:offset` pairs that resolve up to the same address as the one given to you, including the one given, in the `SSSS:OOOO` format. You may output them in any order.

Sample Input

```
D313:BFED
```

Sample Output

```
DF11:000D
DF10:001D
DF0F:002D
DF0E:003D
DF0D:004D
DF0C:005D
DF0B:006D
DF0A:007D
DF09:008D
DF08:009D
DF07:00AD
DF06:00BD
DF05:00CD
DF04:00DD
```

Sample output reduced to save your paper. Please visit https://dmoj.ca/problem/equivaddr for the full sample output.