# B - Reflected Photons

**Input:** `standard input`
**Output:** `standard output`

Jack and Tony are pursuing a suspect through a hall of mirrors. The problem is that if they fire their lasers into the hall of mirrors, who knows where the laser beam is going to exit after it has bounced around for a while? (Actually, the CTU agents don't use lasers, but it was such a cute problem...)

Imagine the hall of mirrors is a square rectangular grid whose entry/exit points are numbered around grid clockwise starting from the upper left. For example,

```
            1111
   1234567890123
  +-------------+
32|             |14
31|      /      |15
30|             |16
  +-------------+
   2222222222111
   9876543210987
```

(In the input file, the grid borders and the numbers will not be present; the above is for illustration only.) A beam of light that enters the grid at 2 will, for example, exit the grid at 28 since it never hits a mirror. However, a beam entering at 31 will reflect off the mirror and exit at 7. All mirrors will be represented by / or by \, meaning that all mirrors are diagonal, thus ensuring that a photon's exit point is different from its entry point.

Given a hall of mirrors, you are to pair off all entry points with its corresponding exit point.

## Input

You will be given a set of input cases, each of which will begin with two unsigned decimal integers (representing the row size and column size of the Hall of Mirrors, each no larger than 20) separated by one space followed by `<EOLN>`. Then will follow the rows of the Hall. Empty squares in the Hall are represented by spaces, mirrors by / or by \. Each row is followed by `<EOLN>`. The last input case will be followed by `0 0<EOLN>`.

## Output

The output cases should appear in the same order as the input cases. Each output case will be of the form `Case c` (where `c` is the number of the input case) followed by `<EOLN>`. followed by the list entry and exit points. Follow the example below for formatting. Note that the entry/exit pairs are sorted both horizontally and vertically and each pair is only listed once. An extra `<EOLN>` follows each output case.

## Sample Input

```
2 3<EOLN>
 / <EOLN>
  \<EOLN>
3 3<EOLN>
   <EOLN>
///<EOLN>
   <EOLN>
0 0<EOLN>
<EOF>
```

## Sample Output

```
Case 1<EOLN>
1<-->8<EOLN>
2<-->10<EOLN>
3<-->5<EOLN>
4<-->7<EOLN>
6<-->9<EOLN>
<EOLN>
Case 2<EOLN>
1<-->11<EOLN>
2<-->9<EOLN>
3<-->8<EOLN>
4<-->12<EOLN>
5<-->7<EOLN>
6<-->10<EOLN>
<EOLN>
<EOF>
```