Problem 6—15-Hole Peg Game
written by David Powers

Bucky is called “Bucky” not only for his buck teeth but also for his skill at Buck-Buck, a game not unlike leapfrog but far more dangerous! A safer variant of leapfrog is the 15-hole peg game.

The 15-hole peg game consists of 15 holes (see diagram below) and 14 pegs. To start the game, a player places 14 pegs in the holes and leaves the 15th hole (their choice) empty. Then moves are made by taking any peg, jumping over another peg and landing in an empty hole. Pegs may be moved in 6 directions, horizontally and diagonally. Each jumped over peg is removed from the board. If a player can make 13 moves or jumps leaving 1 peg on the board, the player wins.

```
                1
               / \
             2  -  3
            / \   / \
          4  -  5  -  6
         / \   / \   / \
       7  -  8  -  9  - 10
      / \   / \   / \   / \
    11 -  12 -  13 -  14 - 15
```

Given the current status of the peg board, determine the set of all possible moves. Note that you are NOT being asked to solve the game, merely to give the set of possible next moves given a specific configuration.

INPUT SPECIFICATION. Each input case consists of a line of unsigned decimal integers between 1 and 15. The line will contain at least one integer; no integer will appear more than once in the line; the integers will be separated by one space and the line terminated by **<EOLN>**. The last case is followed by “–1**<EOLN>**”. This –1 is not to be processed; it merely signifies the end of input. The integers in each line represent the empty holes on the board. All other locations contain pegs.

OUTPUT SPECIFICATION. The output cases should appear in the same order as their corresponding input cases. For each case, you should print “Case *c*” followed by **<EOLN>**. Then you should print for each move “*sp* -> *dp*<**EOLN**>” where *sp* is the source position for a peg and *dp* is the destination position. If there is more than one valid move, you are to order the list so that lower source positions are printed before higher source positions; if there is more than one move from the same source position, you are to order them so that lower destination positions are printed before higher destination positions. An extra **<EOLN>** should follow each output case.

SAMPLE INPUT.

```
3<EOLN>
6·11<EOLN>
-1<EOLN>
<EOF>
```

SAMPLE OUTPUT.

```
Case·1<EOLN>
8·->·3<EOLN>
10·->·3<EOLN>
<EOLN>
Case·2<EOLN>
1·->·6<EOLN>
4·->·6<EOLN>
4·->·11<EOLN>
13·->·6<EOLN>
13·->·11<EOLN>
15·->·6<EOLN>
<EOLN>
<EOF>
```