8th South African Regional
ACM Collegiate Programming Competition

Sponsored by IBM

Problem C – Red balloon
Sudoku

Sudoku is a puzzle played on a 9 by 9 grid. The grid is subdivided into 9 blocks of 3 by 3 cells. At the start of the puzzle, only a few numbers have been recorded on the grid, like this:

9		4		8	7			1
	5					6	8	
2			3					9
	3	6	1		5	2		4
					9		7	
5		7	4				3	8
								5
4		9	6				1	
	1	5			2		6	3

The objective is to complete the grid by filling in numbers in the blank spaces to satisfy the following conditions:

1. Each row must contain each digit 0 through 9 exactly once
2. Each column must contain each digit 0 through 9 exactly once
3. Each 3 by 3 sub-block must contain each digit 0 through 9 exactly once

In general, the solution of a Sudoku puzzle requires backtracking algorithms. A subset of all possible Sudoku puzzles, however, do not require backtracking. These puzzles can be solved by directly addressing the three conditions listed above, and your task will be to solve this kind of Sudoku puzzle.

Input

Your input will consist of an arbitrary number of input records. Each record describes the positions and values of the given cells, representing the starting state of the grid. A record consists of a number of lines, each line containing 5 integer values, formatted as follows:

```
<block_column_nr> <block_row_nr> <cell_column_nr> <cell_row_nr> <digit>
```

where <block_column_nr> denotes the 3x3 sub-block column number, and <cell_column_nr> denotes the cell column number within that sub-block, and likewise for the rows. Note that row and column numbers are numbered starting from 0.

Each input record is terminated by a line containing the value -1, except the last record in the input, which is terminated by a line containing the value -2 instead.

Output

As output, you should print out the completed Sudoku puzzle, in the format shown in the Sample Output section below.

Sample Input

```
0 0 0 0 9
0 0 2 0 4
0 0 1 1 5
0 0 0 2 2
1 0 1 0 8
1 0 2 0 7
1 0 0 2 3
2 0 2 0 1
2 0 0 1 6
2 0 1 1 8
2 0 2 2 9
0 1 1 0 3
0 1 2 0 6
0 1 0 2 5
0 1 2 2 7
1 1 0 0 1
1 1 2 0 5
1 1 2 1 9
1 1 0 2 4
2 1 0 0 2
2 1 2 0 4
2 1 1 1 7
2 1 1 2 3
2 1 2 2 8
0 2 0 1 4
0 2 2 1 9
0 2 1 2 1
0 2 2 2 5
1 2 0 1 6
1 2 2 2 2
2 2 2 0 5
2 2 1 1 1
2 2 1 2 6
2 2 2 2 3
-2
```

Sample Output

```
 9  6  4 | 5  8  7 | 3  2  1
 3  5  1 | 2  9  4 | 6  8  7
 2  7  8 | 3  6  1 | 4  5  9
-------------------------------
 8  3  6 | 1  7  5 | 2  9  4
 1  4  2 | 8  3  9 | 5  7  6
 5  9  7 | 4  2  6 | 1  3  8
-------------------------------
 6  2  3 | 7  1  8 | 9  4  5
 4  8  9 | 6  5  3 | 7  1  2
 7  1  5 | 9  4  2 | 8  6  3
```