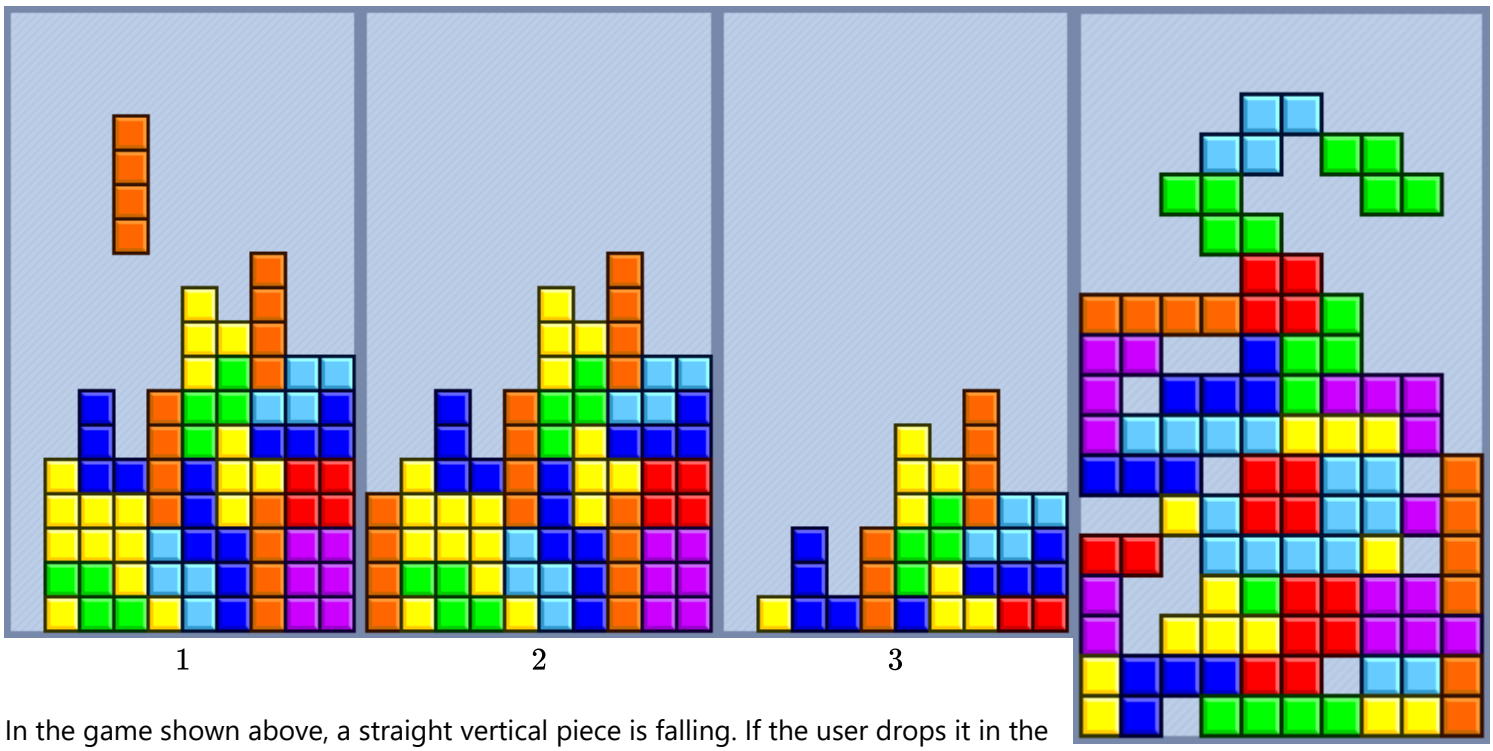
ECOO '13 R2 P1 - Upsetris

Time limit: 2.0s **Memory limit:** 64M

The game of Upsetris is played just like the classic game of Tetris, but with a twist. Just like in Tetris, puzzle pieces, each made of four square bricks, fall from above and the user has to decide where they will land to try and make full rows of blocks. The game ends when the pile reaches the top of the screen.

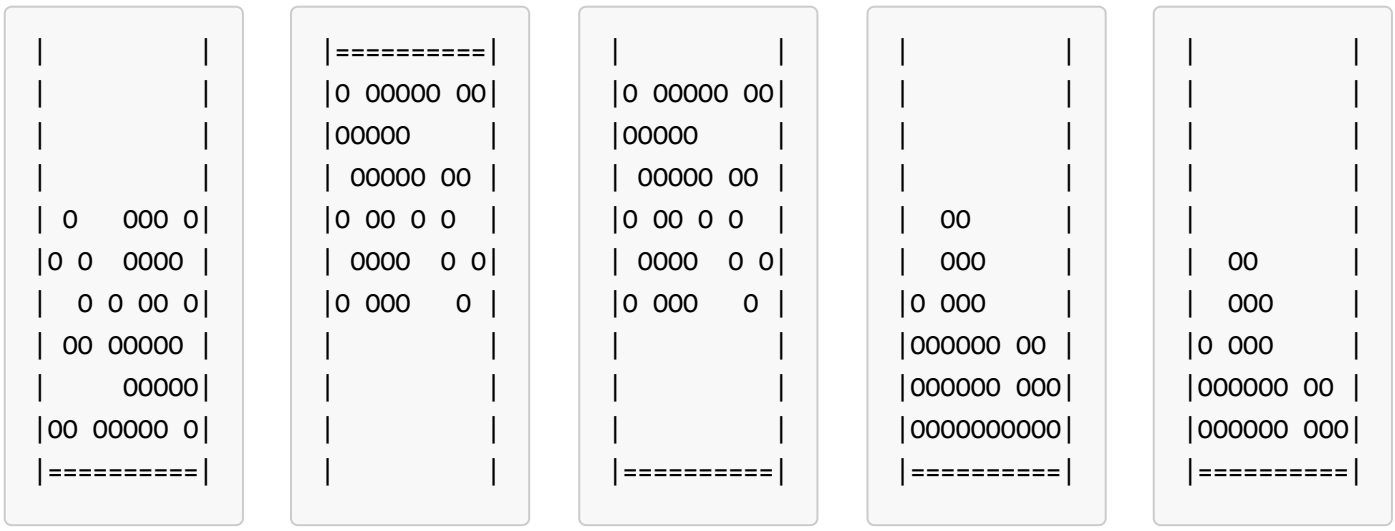


In the game shown above, a straight vertical piece is falling. If the user drops it in the left most column (picture 1), she will make the bottom four rows full (picture 2), and the full rows will disappear allowing the pile to move down (picture 3).

If the player makes bad decisions about where to put pieces, they can end up with boards with "holes" in them as shown at above-right. These holes will eventually cost them the game.

In Upsetris, the "twist" (pun intended) is that the user has a one-time-only chance to "upset" the board by rotating it 180 degrees. When this happens, all the individual bricks separate and fall to the floor. This eliminates the holes and sometimes makes new rows.

Here's an example of the "upset" process in action:



In the example above, the bricks are represented by capital `0`'s and the floor is a row of equal signs. The original board (1) is rotated 180 degrees (2), then the floor of the board is moved to the bottom (3), then the pieces all settle to the new floor (4) making a single complete row which gets removed (5).

The input contains five different Upsetris boards, each with from 5 to 20 rows (not including the "floor") and from 5 to 20 columns (not including the walls). You must simulate the "upset" operation and show the final result for each board. You must show the complete board in your output, including the "floor" row.

Note that all the boards in the sample input are the same size and have the same number of rows as columns, but this might not be true of the data sets you will be judged on. The side characters for the board `|` are ASCII code 124.

Sample Input

```
|           |
|           |
|0          |
|0 00   0 0 |
|     0 0   |
| 0    0 00 |
|  0       00|
|0 0        |
|    0   0 0 |
|0       0  |
|==========|
|           |
|           |
|           |
|     0     |
|    0  00 0|
|    0      0|
|     0  0  |
|        00 |
|0          |
|    0      0|
|==========|
|           |
|           |
|           |
|           |
| 0    000 0|
|0 0   0000 |
|   0 0 00 0|
| 00 00000  |
|      00000|
|00 00000 0|
|==========|
|           |
|           |
|           |
|           |
|           |
|           |
|           |
|           |
| 0      0  |
|      0    0|
|==========|
|           |
|           |
```

```
|           |
|           |
|00  00  0000|
|  0  0000000|
|0  0000  000|
|  00000000  |
|0000  0  0  |
|0   000  000|
|==========|
```

Sample Output

```
|           |
|           |
|           |
|           |
|           |
|           |
|           |
|           |
| 0         0|
| 0 0    0 0|
| 000   00 0|
|==========|
|           |
|           |
|           |
|           |
|           |
|           |
|           |
|           |
|0 0    0   |
|0 0   00   |
|0000 00   0|
|==========|
|           |
|           |
|           |
|           |
|           |
|   00      |
|   000     |
|0 000      |
|000000 00  |
|000000 000|
|==========|
|           |
|           |
|           |
|           |
|           |
|           |
|           |
|           |
|           |
|0 0 0    0 |
|==========|
|           |
|           |
```

```
|           |
|           |
|           |
|           |
|           |
|   0    0  |
|  00 000   |
|000 000 00 |
|===========|
```