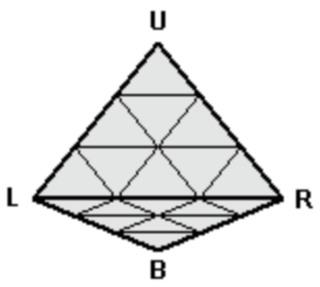
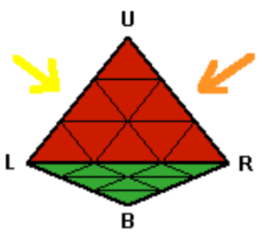
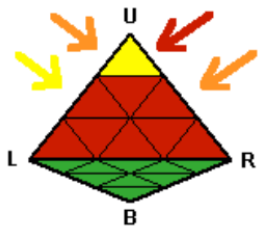
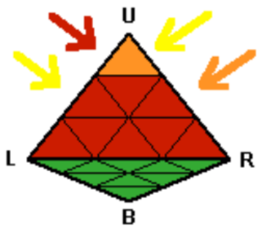
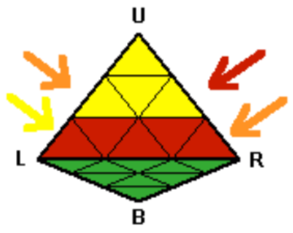
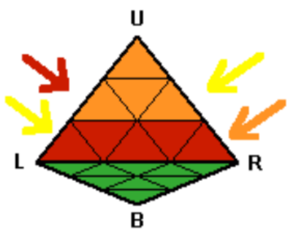
You've mastered the Rubik's Cube and got bored solving it, so now you are looking for a new challenge. One puzzle similar to the Rubik's Cube caught your attention. It's called a *Pyraminx puzzle*, and is a triangular pyramid-shaped puzzle. The parts are arranged in a pyramidal pattern on each side, while the layers can be rotated with respect to each vertex, and the individual tips can be rotated as well. There are `4` faces on the Pyraminx. The puzzle should be held so that one face faces you and one face faces down, as in the image below. The four corners are then labeled U (for up), R (for right), L (for left), and B (for back). The front face thus contains the U, R, and L corners.

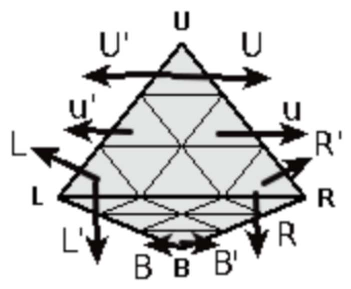


Let's write down all possible moves for vertex `U` in the following notation:

- `U` - `120°` counterclockwise turn of topmost tip (assuming that we're looking at the *pyraminx* from the top, and vertex `U` is the topmost);
- `U'` - clockwise turn for the same tip;
- `u` - `120°` counterclockwise turn of two upper layer;
- `u'` - clockwise turn for the same layers.

Initial state	Move U	Move U'	Move u	Move u'
				

For other vertices the moves can be described similarly:



The first puzzle you bought wasn't assembled, so you get your professional *pyraminx* solver friend to assemble it. He does, and you wrote down all his moves notated as described above. Now the puzzle's faces have colors `faceColors[0]` (front face), `faceColors[1]` (bottom face), `faceColors[2]` (left face), `faceColors[3]` (right face). You want to know the initial state of the puzzle to repeat your friend's moves and see how he solved it.

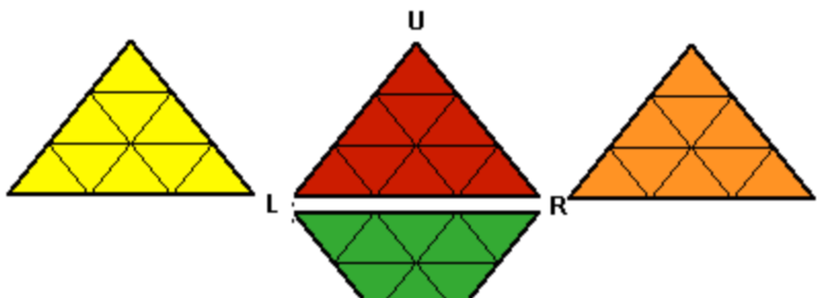
Example

For `faceColors = ['R', 'G', 'Y', 'O']` and `moves = ["B", "b'", "u'", "R"]`, the output should be

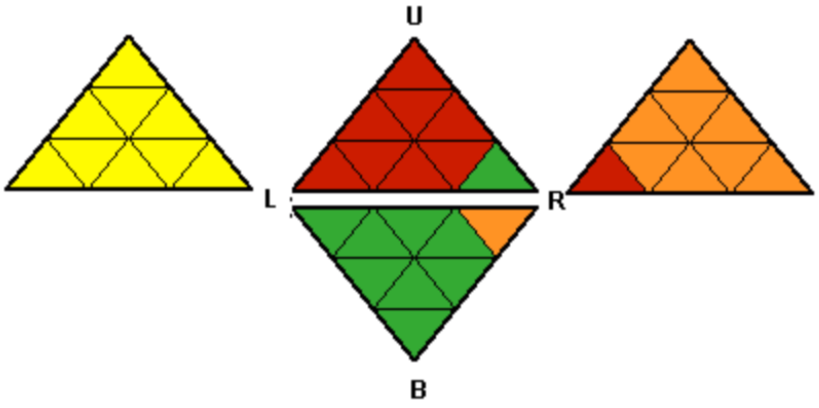
```
pyraminxPuzzle(faceColors, moves) = [['Y', 'Y', 'Y', 'Y', 'R', 'R', 'R', 'R', 'G'],
                                      ['G', 'R', 'O', 'O', 'O', 'G', 'G', 'G', 'G'],
                                      ['Y', 'O', 'Y', 'G', 'O', 'O', 'G', 'G', 'Y'],
                                      ['R', 'O', 'O', 'R', 'O', 'Y', 'Y', 'R', 'R']]
```

Let's repeat the friend's steps in reverse order:

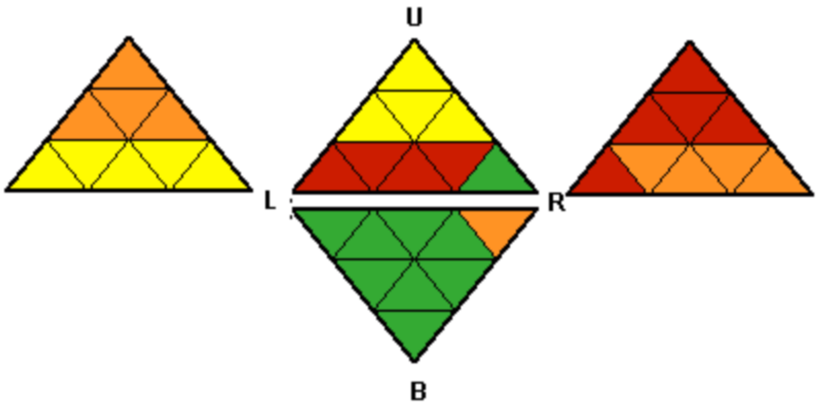
Final state:



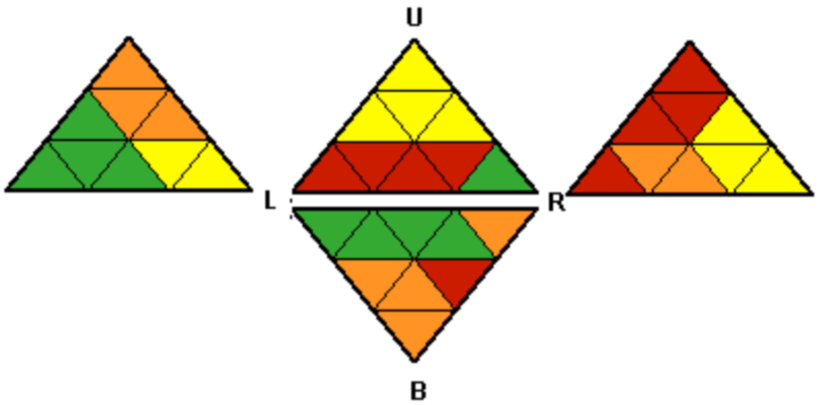
Before the last move:



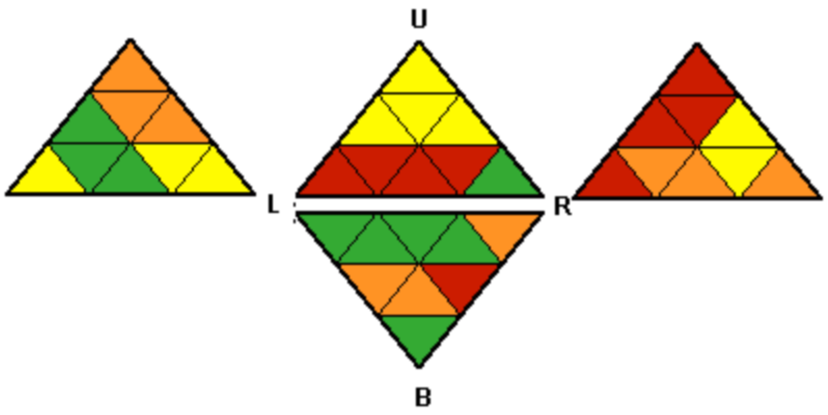
One more move before that:



And one more:



Finally, the initial state:



Input/Output

- **[execution time limit] 4 seconds (py3)**
- **[input] array.char faceColors**

  A distinct array of four distinct characters, representing the front, bottom, left and right faces, respectively.

  Guaranteed constraints:

  `faceColors.length = 4`.

- **[input] array.string moves**

  Guaranteed constraints:

  `1 ≤ moves.length ≤ 100`,

  ```
  moves[i] ∈ {"U", "U'", "u", "u'", "L", "L'", "l", "l'",
              "R", "R'", "r", "r'", "B", "B'", "b", "b'"}.
  ```

- **[output] array.array.char**

  Initial state of the puzzle. `result[0]` should contain `9` characters corresponding to the front face, `result[1]` - to the bottom face, `result[2]` - to the left face and `result[3]` - to the right face.

The colors for each face should be given in top-to-bottom and left-to-right order, starting from the marked vertex (i.e. `U`, `B`, `L` or `R`), assuming that this vertex is at the top of the puzzle.