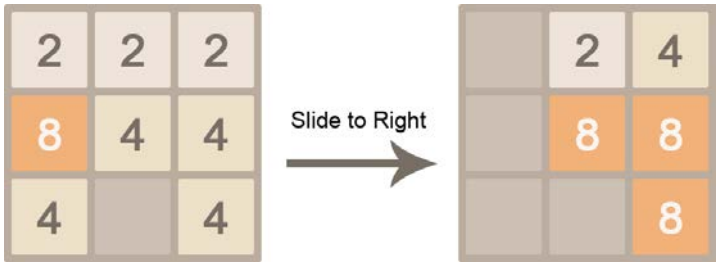
Easy 2048

Time Limit: 10 Seconds Memory Limit: 262144 KB

Have you heard or played the game **2048**? It is a simple and addictive game. The standard **2048** game consists of a grid of 4 rows and 4 columns, with some numbers tiled in the grid. It is a turn-based single player board game. In each turn, the player can use arrow keys to move the tiles slide towards one direction (Up, Down, Left, Right). If two tiles with the same number touch, they will merge into one.

Bob thinks **2048** is too hard for most of people. So he made a easy version called **64**. The rules of **64** is very similar to the rules of **2048**. The only difference is the game is played on a 3 x 3 board and the player needs to get a tile of 64. Before he publish his own game, he need to check the game difficulty to ensure the game won't be too hard or too boring for most of people. Therefore, he ask you to write a program to find the shortest operation sequence to win the game.



Here is a detailed game mechanics:

- At the beginning of the game, the grid contains several (or zero) tiles. Each tile has a number written on it. The other places of the grid is empty.
- During the i-th turn, the player must move the tiles slide to one direction. When the tiles start to slide, all tiles will simultaneously move towards the selected direction with the same speed. A tile will stop if it touches the border of grid or other tiles with different number. But if a tile catch up another tile with the same number, the two tiles will merge into one, with the number equal to the sum of their old values. The newly generated tiles can not merge again during current slide operation.
- If no tiles will move or merge under a selected direction, the direction is considered invalid. The player must choose a valid direction to move, otherwise he will lose. After the slide, the player will win the game if there is at least one tile with number 64.
- The game has a internal random number generator. At the beginning of the i-th turn (except for the first turn), a new pseudorandom values is

generated by $R_i = R_{i-1} * P + Q$. The seed R_1, the multiplier P and the increment Q are preseted values in the game source code.

- At the end of i-th turn, a tile with random number will appear on a random empty cell. The cell is selected by $X = 1 + (R_i \text{ Mod } Count)$. On the X-th empty cell, a tile with number 2 will appear if R_i is not a multiple of 10, otherwise number 4 appears. The value *Count* is the number of empty cells. The cells are ordered by row-major order from top to bottom. In each row, the cells are ordered from left to right.

Input

There are multiple test cases. The first line of input is an integer T indicates the number of test cases. For each test case:

The first line contains three integers R_1, P, Q (0 <= R_1, P, Q <= 1000000). Each of the next 3 lines contains 3 numbers indicates the initial state of the game. The number on the tiles will be one of 2, 4, 8, 16 or 32. Specially, number 0 means there is no tile in this place.

Output

For each test case, output the shortest operation sequence to win the game. The sequence consists of characters of 'U', 'D', 'L', 'R' (stands for Up, Down, Left and Right respectively). Among all possible solutions, you should choose the one with smallest lexicographical order. If the solution does not exist, output "Always lose!" instead.

Sample Input

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

Sample Output

```
DDDLDDDRDLDLRLDDDUDLDLDRDLDDDRDLRDLL
Always lose!
```