Problem 27: Tumbling Die

Source filename:	`die.(cpp|java)`
Input filename:	`die.in`
Output filename:	`die.out`

The 7x7 array below depicts a maze where each square is either equal to the face of an ordinary die or is an asterisk, '*'. To solve the maze one ordinary die is needed to traverse the maze. The die is placed on the middle square (in the test case illustrated in Figure 1 this square has the value 5) with a given initial orientation. For the sake of illustration, let's suppose that the initial orientation of the die has the value 6 on top and the value 4 facing the bottom of the maze. The arrangement of the 6 faces of an ordinary die is shown in the figure 2.

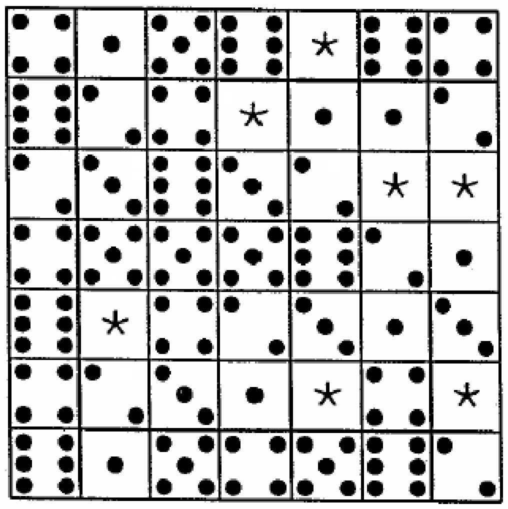

Figure 1

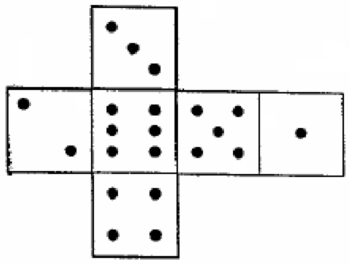

Figure 2

Now, try thinking of the die as a carton that is too heavy to slide onto the next square. You can, however, tip it over on an edge and have it land on an adjacent square. In that manner you move the die horizontally or vertically from one square to another. There is also the restriction that you can only move the die onto a square that contains the same number as the number on top of the die before it is tipped over. However, a square with a star (or asterisk) is a "wild" square and you may always tip the die to one of these squares no matter what number is on top of the die.

For example, when we first begin with the illustrated maze above, the only possible 1st move is to tip the die to the right, onto the square with the six. When we tip the die onto that square, the 2 will appear on top of the die. We can now either tip up one square or to the right one square since both of these squares show a value of 2. Suppose that we tip the die to the right. The value 1 will now appear on top of the die. At this point we have three choices: tip the die to the right, tip the die downward, or tip it up to the "wild" square.

The center square is not only the starting point, but is also the goal. To solve the maze, you must tip the die off of the center square, and then find a way to move it back to the center square by following the "tipping" rules outlined above.

The input file contains several test cases. The 1st line of each test case contains 3 non-negative integers: *n*, *top*, *side*, all separated by a single space. The integer *n* denotes the size of the square maze. If this integer is 0, you have reached the end of the input file. Otherwise, *n* will be an **odd** integer between 3 and 99, inclusive. The integer *top* ($1 \le top \le 6$) represents the side of the die that is on top in the initial orientation of the die as it is placed on the middle square. The integer *side* ($1 \leq side \leq 6$) represents the side of the die that is facing towards the bottom of the maze in the initial orientation of the die.

The next *n* lines represent the *n* rows of the square maze. Each of these rows will contain *n* characters each of which will be either a digit between 1 and 6, inclusive, or an asterisk.

For each such test case the output file should contain on one line a list of moves that solves the maze. The list of moves should be output using the letters ‘`R`’, ‘`L`’, ‘`U`’, & ‘`D`’ which represent the directions ‘Right’, ‘Left’, ‘Up’, & ‘Down’.

There may be more than one possible solution. If such is the case, you are free to choose any solution. If no solution is possible, you should output the phrase: “`No Solution!`”.

There should be NO spaces in the output file.

<u>Example Input File</u> <u>(dice.in)</u>

```
3 6 4
632
562
463
3 6 4
632
556
423
7 6 4
4156*64
624*112
23632**
4555621
6*42313
4231*4*
6154562
0 0 0
```

<u>Example Output File</u> <u>(dice.out)</u>

```
DRUL
No Solution!
RRRULURDLLDDDRRUUULLUULLLDDDDDRRDLLUUUULUURDRRDD
```