Problem: Xonix

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds for 10 moves
Memory limit:	256 megabytes

> First version of Xonix was created in 1984 as a Qix clone, which appeared earlier on arcade automates.
>
> wikipedia

In this problem you are offered to play "Xonix"modification for two players. The game is conducted on 15×15 field. Each player has a "cursor which on every move can be moved in one of four directions: left, up, right or down. After cursor movement, the cell, which it now occupies are added to player's territory. Also all cells, which are surrounded by player's territory, are added to his territory. Cell is considered surrounded by player if you are not able to reach field border from that cell without going by that player's cell.

Let us consider a small example. Let us denote first player's territory by `1`, second player's territory by `2`, first player's cursor by `3`, second player's cursor by `4`. Free cells we will denote by `.`.

Consider the following situation:

```
1 1 1 1
1 . . .
1 . . 3
1 1 1 1
```

If first player will move cursor up, he will add four additional cells to his territory, so the field will look like following:

```
1 1 1 1
1 1 1 3
1 1 1 1
1 1 1 1
```

But in such situation:

```
1 . . .
1 . . 3
1 . . 1
1 1 1 1
```

After first player's move up, he will not get additional cells to his territory, because you can reach field border without going by first player's cells. The field will look like that:

```
1 . . 3
1 . . 1
1 . . 1
1 1 1 1
```

If after player's move there are cells, from which you can't reach field border without using that player's cells, but possible to reach field border using other player's cells, such cells are not considered surrounded.

Player can not make move to other player's territory.

Game ends when there is no free cells in the field or after 1000 moves (500 moves for each player).

Input

You will receive a map before each move. Map consists of 15 lines with 15 characters in each. Every character denotes one cell. Free cells are denoted by `'.'`, your territory is denoted by `1`, your opponent's territory is denoted by `2`, your cursor is denoted by `3`, your opponent's cursor is denoted by `4`.

Output

Output one character for every move. This character denotes direction, which you want to move your cursor in. Move left (decreasing column) is denoted by `L`, move right (increasing column) is denoted by `R`, move up (decreasing row) is denoted by `U`, move down (increasing row) is denoted by `D`.

Output an empty line after your move (so you need to output direction character and two newlines).

If your move is incorrent (moving to your opponent's territory or outside the field), it will be ignored.

Don't forget to flush output buffer after every move.

Please remember, that your program should not terminate by itself, so make a dummy read operation after last move.

Example shows output for one move:

Examples

standard input
3............. 4
standard output
D

Sample program in C++:

```
#include <iostream>
#include <fstream>
#include <ctime>
#include <cstdlib>
#include <vector>

using namespace std;

const int SZ = 15;

bool Good(char c)
{
    if (c == '.' || c >= '1' && c <= '4') return true;
    return false;
}

const char moves[5] = "LURD";
char field[SZ][SZ];

int main(int argc, char *argv[])
{
    srand((unsigned)time(NULL));
    int we = atoi(argv[1]);
    char enemyhive = (char)(3 - we + '0');
    int i, j;
    while (1)
    {
        for (i = 0; i < SZ; i++)
        {
            for (j = 0; j < SZ; j++)
            {
                char c = getchar();
                while (!Good(c)) c = getchar();
                field[i][j] = c;
            }
        }
        printf("%c\n", moves[rand() % 4]);
        printf("\n");
        fflush(stdout);
    }
    return 0;
}
```