Problem: Tic-Tac-Toe

Input file:	`standart input`
Output file:	`standart output`
Time limit:	2 seconds for 10 moves
Memory limit:	64 megabytes

You are to implement a strategy for playing "Tic-Tac-Toe"game on board of size 20 × 20.

Let us remind the game rules: two players make moves in turns. On his turn, player chooses any free cell on the board and puts there his own symbol (cross for first player and nought for second). Player who manages to put five symbols in a row is declared a winner. If no one manages to do so until there is no free cell, game is considered a draw.

In this problem round system will be used for ranking players. Your solution will play twenty games with each other solution: ten games as the first player and ten games as the second. Your solution will get 2 points for every winned game and 1 points for every draw. You will not get points for losing.

Input

You will receive current field status in stdin before each move. You will get 20 lines with 20 numbers in each. Field status will be ended by a blank line. Every number corresponds to one cell of the field. The number is 0 for free cell, 1 if there's a cross in a cell and 2 if there's a nought in the cell.

You can determine if you are first or second player by command-line argument which your program will receive from testing system. If you receive `1` then you're the first player, otherwise you're the second. Also you can just examine the field first time you've received it. If you received an empty field for the first time, you're the first, otherwise the second.

Output

For every move you're to output row and column number of the cell you want to place your symbol. Rows of the field are numbered top-bottom beginning from 0. Columns are numbered from left to right, also beginning from zero. Separate row and column number with space. Don't forget to output an empty line after each move (see sample programs for clarification). Also don't forget to flush output buffer after move (see sample programs).

Examples

standart input

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

standart output

```
1 10
```

Example above represents field status which you will receive before each move. In this sample it's second player's turn.

Sample program for C++ (makes random move each turn):

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

using namespace std;

const int SZ = 20;
int field[SZ][SZ];

int main()
{
    srand((unsigned)time(NULL));
    int i, j;
    while (1)
    {
        for (i = 0; i < SZ; i++)
        {
            for (j = 0; j < SZ; j++)
            {
                scanf("%d", &field[i][j]);
            }
        }
        while (1)
        {
            int ni, nj;
            ni = rand() % SZ;
            nj = rand() % SZ;
            if (field[ni][nj] == 0)
            {
                printf("%d %d\n\n", ni, nj);
                break;
            }
        }
        fflush(stdout);
    }
    return 0;
}
```

Sample program for Pascal (makes random move each turn):

```
const SZ = 20;
var field : array[1..SZ, 1..SZ] of integer;
i, j, ni, nj : integer;
begin
randomize;
while true do begin
        for i := 1 to SZ do begin
         for j := 1 to SZ do begin
         read(field[i, j]);
         end;
        end;
        while true do begin
            ni := trunc(random() * 1000) mod SZ + 1;
            nj := trunc(random() * 1000) mod SZ + 1;
            if field[ni][nj] = 0 then begin
             writeln(ni, ' ', nj);
             writeln;
             break;
            end;
        end;
        flush(output);
    end;
end.
```