PROBLEMS | SUBMIT CODE | MY SUBMISSIONS | STATUS | HACKS | ROOM | STANDINGS | CUSTOM INVOCATION

A. Beautiful Matrix

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You've got a 5×5 matrix, consisting of 24 zeroes and a single number one. Let's index the matrix rows by numbers from 1 to 5 from top to bottom, let's index the matrix columns by numbers from 1 to 5 from left to right. In one move, you are allowed to apply one of the two following transformations to the matrix:

1. Swap two neighboring matrix rows, that is, rows with indexes i and $i + 1$ for some integer i $(1 \le i < 5)$.
2. Swap two neighboring matrix columns, that is, columns with indexes j and $j + 1$ for some integer j $(1 \le j < 5)$.

You think that a matrix looks *beautiful*, if the single number one of the matrix is located in its middle (in the cell that is on the intersection of the third row and the third column). Count the minimum number of moves needed to make the matrix beautiful.

Input

The input consists of five lines, each line contains five integers: the j-th integer in the i-th line of the input represents the element of the matrix that is located on the intersection of the i-th row and the j-th column. It is guaranteed that the matrix consists of 24 zeroes and a single number one.

Output

Print a single integer — the minimum number of moves needed to make the matrix beautiful.

Examples

input Copy

```
0 0 0 0 0
0 0 0 0 1
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
```

output Copy

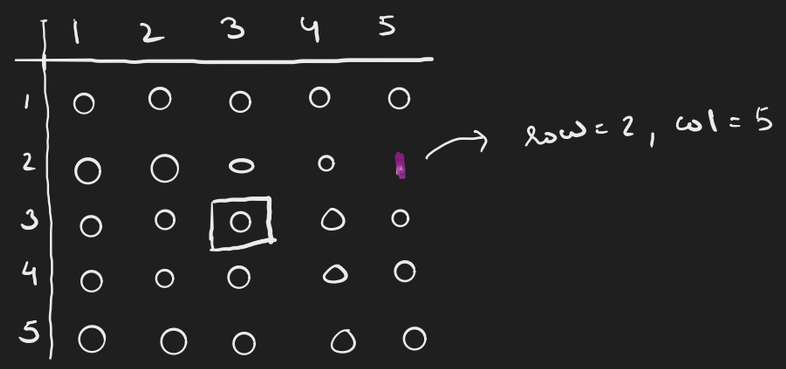
```
3
```

input Copy

```
0 0 0 0 0
0 0 0 0 0
0 1 0 0 0
0 0 0 0 0
0 0 0 0 0
```

output Copy

```
1
```



Given :

Beautiful Matrix
- mat[3][3] = 1.
- Rest all cells = 0.

Goal :

→ Min no. of moves to make matrix a beautiful one.

```
Solve (matrix) {
    int row, col;

    for (i ← 0 to 4) {
        for (j ← 0 to 4) {
            if (matrix[i][j] == 1) {
                row = i+1, col = j+1;
                break;
            }
        }
    }

    return abs(3 - row) + abs(3 - col);
}
```

1. Initialize 2 variables
   → row, col.
2. Traverse the matrix.
3. if mat[i][j] == 1
   → row = i+1, col = j+1
   → Terminate traversal.
4. Return abs(3 - row)
   + abs(3 - col).