Mirrored game

Ivancho learned a new game. It is played on two wooden boards put one above the other. Each boards is divided into a table containing **N** rows and **N** columns. In each cell on the boards there is a chip - eigther white or black.

During the game, every turn you can take a chip from a cell from the first board and a chip from its corresponding mirrored cell, and swap them. A mirrored cell is a cell which is placed in the same column, but on the mirrored row. The first row on the first table is mirror to the last row on the second table, the second row on the first table is mirrored to the row before the last on the second table, etc.

The goal of the game is to move all black chips to a single board. Ivancho wants to become a great player, so he wants to learn to do that within as little turns as possible.

Help him by writting a game **chips**, which by a givven starting condition of the board, finds the minimum number of turns required to move all black chips to one of the boards.

Input: the first line of the input file **chips.in** will contain a single integer **N** - the number of rows and columns on each board.
After that N lines follow, each containg N zeros or ones, which describe a row from the first table. A zero means that a white chip, a one means a black chip.
After that N lines follow, each containg N zeroes or ones. They discribe a row from the second board.

Output: The output file chips.out should contain a single integer - the minimum turns required to move all black chips to a single board, if that is possible, or -1, if it is not.

Limits:
2 <= N <= 30

Time limit: 0.2 sec.
Memory limit: 256 MiB.

Remark: Reading and writing to a file can be done using the appropriate statement. You can use the *freopen* statement from the standard library *cstdio* by adding the flowing two lines at the beginning of your main function:

```
    freopen ("chips.in", "r", stdin);
    freopen ("chips.out", "w" , stdout);
```

Number of preliminary tests: 4
Number of final tests: 10

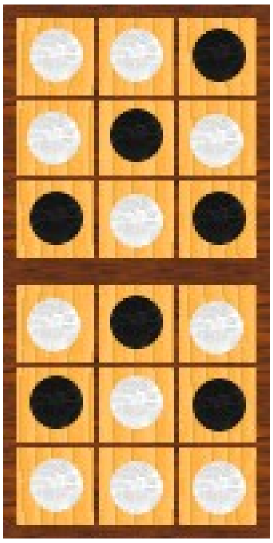
Example test:

chips.in	chips.out

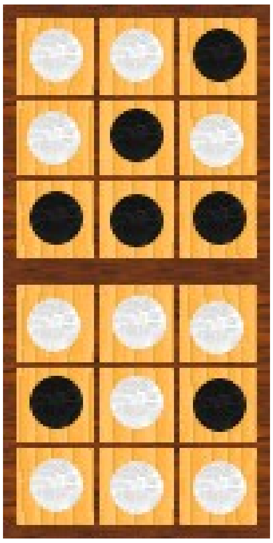
3
0 0 1
0 1 0
1 0 1
0 1 0
1 0 1
0 0 0
	3

Output explanation:
Here is one possible solution:

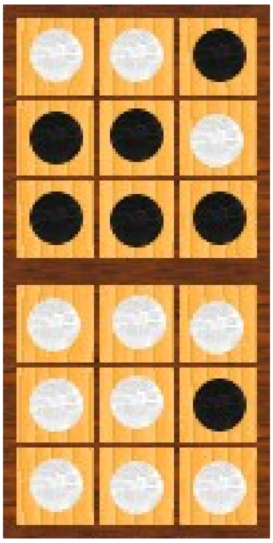
Initial:



After turn 1:



After turn 2:



After turn 3: