Problem ID: gamehistorian

You are travelling through ancient Rome in the first century BC. You notice a lot of chalk markings on the walls that you quickly identify as tic-tac-toe games. As you can recall from a history lesson, the Romans actually invented the popular game. You get really excited about the possibility to interact with people from this time (and to beat them in a game to show your intellectual superiority).

It is common knowledge that, if played properly, it is always possible to end a game of tic-tac-toe in at least a draw. However, on your way to your first opponent, you get nervous. What if you make a wrong move and suddenly lose? That would be extremely embarrassing. So instead of trusting your own abilities, you decide to create a program that will never lose (ties are fine). Fortunately, you have already convinced your waiting opponents to let you make the first move in all games, which should give you an advantage.

As a reminder, here are the rules of tic-tac-toe:

- Two players take turns marking the spaces in a 3×3 grid (with the first turn being yours).
- Your mark is an `X`, your opponent's is an `O`.
- A player chooses exactly one (as of yet unmarked) space during their turn.
- As soon as a row, column, or diagonal contains three `X` marks, the game ends and you win.
- As soon as a row, column, or diagonal contains three `O` marks, the game ends and you lose.
- If all nine spaces are marked but neither player's win condition applies, the game ends in a tie.

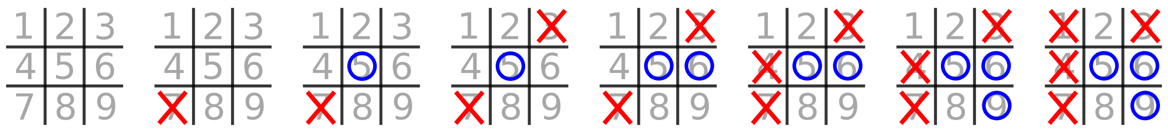

Figure 1: Visualisation of the given sample.

Interaction Protocol

Your submission will be interacting with a special program called the *grader*. This means that the output of your submission is sent to the grader and the output of the grader is sent to the standard input of your submission. This interaction must follow a specific protocol:

Your submission and the grader alternate writing moves, with your submission going first. A move is a line with a single integer s ($1 \le s \le 9$) specifying the space marked by the player, corresponding to the numbering in Figure 1. You can safely assume that the grader will only perform valid moves.

After every move you should *flush* the standard output to ensure that the move is sent to the grader. For example, you can use `fflush(stdout)` in C++, `System.out.flush()` in Java, `sys.stdout.flush()` in Python, `hFlush stdout` in Haskell, and `CALL FLUSH()` in Fortran.

The game ends as soon as one of the two players manages to complete a row/column/diagonal or all cells have been marked. After the game ends, your submission should terminate with exit code `0` as usual. For convenience, the grader will send a single `-1` to signal the end of the game, which your submission may or may not read.

Your submission will be accepted if it follows the protocol above and achieves a win or a tie. If it makes an illegal move (i.e. writes anything other than a cell number or tries to move to a cell that has already been marked), it will be judged as “Wrong Answer”.

Note that for testing purposes, your submission will still be run multiple times, and the grader will not make the same moves each time.

Sample Input 1

```

5

6

9

-1
```

Sample Output 1

```
7

3

4

1
```

In the example interaction above, one possible output of the grader is on the left and one possible correct output of a submission is on the right.
The empty lines on both sides only serve to emphasise the chronological order of requests and answers. The grader will never output any empty lines.