Demonstration Task 2: Game Show

In a TV game show, Jack and Jill are placed in separate rooms where each is given a number between 1 and N.

Each room has a red button that lights a red light in the other room, and a green button that lights a green light in the other room.

Jack goes first. He presses either the green or the red button in his room which lights the corresponding light in Jill's room. Then it is Jill's turn to press one of her buttons, and so on.

Jack and Jill have a common objective: to determine whether or not they have been given the same number. At any time, either Jack or Jill may call out *same!* or *different!*, which terminates the game.

Your task is to implement two program components, `jack` and `jill` that together will play the game. The master of ceremonies is implemented by the program component `gameshow` which is supplied by IOI.

Jack must implement two procedures: **jackstart(N,K)** and **jacksturn(C)**. Jill must implement two procedures: **jillstart(N,L)** and **jillsturn(C)**. **jackstart(N,K)** will be called by the grader exactly once at the start of the game, specifying **N** as described above and Jack's secret number **K**, which will be between 1 and N. **jillstart(N,L)** will similarly be called at the start of the game, indicating N and Jill's secret number **L**. Then **jacksturn(C)** and **jillsturn(C)** will be called alternately, starting with **jacksturn**

The parameter C is an integer indicating the color of button pressed by the other player: C=0 indicates "none" and is used only for the first call to **jacksturn(C)**. C=1 indicates "green" while C=2 indicates "red."

jacksturn and **jillsturn** should return:

- 1 to press the green button
- 2 to press the red button
- 3 if Jack or Jill has determined that K and L are equal
- 4 if Jack or Jill has determined K and L are not equal

Note: Jack and Jill must communicate only through the specified interface. Shared variables, file access and network access are prohibited. In C or C++, you may declare persistent variables to be `static` *to retain information for Jack or Jill, while preventing them from being shared. In Pascal, you may declare persistent variables in the* `implementation part` *of your solution files.*

Subtask 1 [25 points]

Implement any correct strategy for N=10. *The implementation files described below contain a solution to this subtask.*

Subtask 2 [25 points]

Implement any correct strategy for N=10 that yields the correct answer with no more than 10 calls to **jacksturn** and **jillsturn**, in total.

Subtask 3 [25 points]

Implement any correct strategy for N=10 that yields the correct answer with no more than 5 calls to **jacksturn** and **jillsturn**, in total.

Subtask 4 [25 points]

Implement any correct strategy for N≤1,000,000,000 that runs within the time limit of 10 seconds.

Implementation Details

- Use the RunC programming and test environment
- Implementation folder: `/home/ioi2010-contestant/gameshow/` (download prototype here)
- To be implemented by contestant:
  - `jack.c` or `jack.cpp` or `jack.pas`
  - `jill.c` or `jill.cpp` or `jill.pas`
- Contestant interface:
  - `jack.h` or `jack.pas`
  - `jill.h` or `jill.pas`
- Grader interface: *none*
- Sample grader: `grader.c` or `grader.cpp` or `grader.pas`
- Sample grader input: `grader.in.1` `grader.in.2` ...
  Note: the sample grader reads N, K, L from standard input.
- Expected output for sample grader input: `grader.expect.1` `grader.expect.2` `...`
- Compile and run (command line): `runc grader.c` or `runc grader.cpp` or `runc grader.pas`
- Compile and run (gedit plugin): *Control-R*, while editing any implementation or grader file.
- Submit (command line): `submit grader.c` or `submit grader.cpp` or `submit grader.pas`
- Submit (gedit plugin): *Control-J, while editing any implementation or grader file.*
- *CPU time limit: 10 seconds*
- *Memory limit: 256 MB*