Day 1 Task 1: Cluedo

Dr. Black has been murdered. Detective Jill must determine the murderer, the location, and the weapon. There are six possible murderers, numbered 1 to 6. There are ten possible locations, numbered 1 to 10. There are six possible weapons, numbered 1 to 6.

For illustration only, we show the names of the possible murderers, locations and weapons. The names are not required to solve the task.

Murderer	Location	Weapon
1. Professor Plum	1. Ballroom	1. Lead pipe
2. Miss Scarlet	2. Kitchen	2. Dagger
3. Colonel Mustard	3. Conservatory	3. Candlestick
4. Mrs. White	4. Dining Room	4. Revolver
5. Reverend Green	5. Billiard Room	5. Rope
6. Mrs. Peacock	6. Library	6. Spanner
	7. Lounge	
	8. Hall	
	9. Study	
	10. Cellar	

Jill repeatedly tries to guess the correct combination of murderer, location and weapon. Each guess is called a *theory*. She asks her assistant Jack to confirm or to refute each theory in turn. When Jack confirms a theory, Jill is done. When Jack refutes a theory, he reports to Jill that one of the murderer, location or weapon is wrong.

You are to implement the procedure **Solve** that plays Jill's role. The grader will call **Solve** many times, each time with a new case to be solved. **Solve** must repeatedly call **Theory(M,L,W)**, which is implemented by the grader. M, L and W are numbers denoting a particular combination of murderer, location and weapon. **Theory(M,L,W)** returns 0 if the theory is correct. If the theory is wrong, a value of 1, 2 or 3 is returned. 1 indicates that the murderer is wrong; 2 indicates that the location is wrong; 3 indicates that the weapon is wrong. If more than one is wrong, Jack picks one arbitrarily between the wrong ones (not necessarily in a deterministic way). When **Theory(M,L,W)** returns 0, **Solve** should return.

Example

As example, assume that Miss Scarlet committed the murder (Murderer 2) in the conservatory (Location 3) using a revolver (Weapon 4). When procedure **Solve** makes the following calls to function **Theory**, the results in the second column could be returned.

Call	Returned value	Explanation
Theory(1, 1, 1)	1, or 2, or 3	All three are wrong
Theory(3, 3, 3)	1, or 3	Only the location is correct
Theory(5, 3, 4)	1	Only the murderer is wrong
Theory(2, 3, 4)	0	All are correct

Subtask 1 [50 points]

Each test run may call **Solve** up to 100 times. Each call might correspond to a different combination of murderer, location and weapon as the answer. Each time **Solve** is called, it must find the correct theory with no more than 360 calls to **Theory(M,L,W)**. *Be sure to initialize any variables used by **Solve** every time it is called.*

Subtask 2 [50 points]

Each test run may call **Solve** up to 100 times. Each time **Solve** is called, it must find the correct theory with no more than 20 calls to **Theory(M,L,W)**. *Be sure to initialize any variables used by **Solve** every time it is called.*

Implementation Details

- Implementation folder: `/home/ioi2010-contestant/cluedo/`
- To be implemented by contestant: `cluedo.c` or `cluedo.cpp` or `cluedo.pas`
- Contestant interface: `cluedo.h` or `cluedo.pas`
- Grader interface: `grader.h` or `graderlib.pas`
- Sample grader: `grader.c` or `grader.cpp` or `grader.pas` *and* `graderlib.pas`
- Sample grader input: `grader.in.1`.
  Note: Each line of input contains three numbers denoting the murderer, the location and the weapon.
- Expected output for sample grader input: if **Solve** correctly solves all cases, the output file will contain `OK t` where `t` is the maximum number of calls to **Theory** used for any case.

- 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 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.