Interactive Treasure Hunt

Input file: `standard input`
Output file: `standard output`
Time limit: 3 seconds
Memory limit: 512 megabytes

This is an interactive problem.

There is a grid of $n \times m$ cells. Two treasure chests are buried in two different cells of the grid. Your task is to find both of them. You can make two types of operations:

- `DIG` r c: try to find the treasure in the cell (r, c). The interactor will tell you if you found the treasure or not.
- `SCAN` r c: scan from the cell (r, c). The result of this operation is the sum of Manhattan distances from the cell (r, c) to the cells where the treasures are hidden. Manhattan distance from a cell (r_1, c_1) to a cell (r_2, c_2) is calculated as $|r_1 - r_2| + |c_1 - c_2|$.

You need to find the treasures in at most 7 operations. This includes both `DIG` and `SCAN` operations in total. To solve the test you need to call `DIG` operation at least once in both of the cells where the treasures are hidden.

Interaction Protocol

Your program has to process multiple test cases in a single run. First, the testing system writes t — the number of test cases ($1 \le t \le 100$). Then, t test cases should be processed one by one.

In each test case, your program should start by reading the integers n and m ($2 \le n, m \le 16$).

Then, your program can make queries of two types:

- `DIG` r c ($1 \le r \le n$; $1 \le c \le m$). The interactor responds with integer 1, if you found the treasure, and 0 if not. If you try to find the treasure in the same cell multiple times, the result will be 0 since the treasure is already found.
- `SCAN` r c ($1 \le r \le n$; $1 \le c \le m$). The interactor responds with an integer that is the sum of Manhattan distances from the cell (r, c) to the cells where the treasures were hidden. The sum is calculated for both cells with treasures, even if you already found one of them.

After you found both treasures, i. e. you received 1 for two `DIG` operations, your program should continue to the next test case or exit if that test case was the last one.

Example

standard input	standard output
`1`	
`2 3`	
	`SCAN 1 2`
`1`	
	`DIG 1 2`
`1`	
	`SCAN 2 2`
`3`	
	`DIG 1 1`
`0`	
	`DIG 1 3`
`1`	