Problem F. Four Colors

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

This is an interactive problem.

Fred and Fiona are tired of playing tic-tac-toe at boring lessons, so they have invented the new game. The board for the game is a connected undirected graph without cycles, also known as a tree. Players alternate their turns, Fred moves first. The players use pencils of four colors for the game: red, green, blue and yellow, we will denote colors by integers from 1 to 4.

Initially all vertices of the tree are uncolored. Each move the current player chooses some uncolored vertex and colors it with one of the four possible colors in such way that no two vertices connected by an edge have the same color.

If the current player has no moves because all vertices are colored already, or there is no vertex that can be correctly colored, the game ends and the winner is determined in the following way. If all vertices are colored, Fred wins, if there is an uncolored vertex, Fiona wins.

Friends have played many games, and Fiona has noticed that Fred always wins. After a thought she understood that there is a winning strategy for Fred that he has found. Can you do the same?

Interaction Protocol

Your program will play the game with judges' program via standard input and output.

First your program must read the description of the tree. If is specified by n — the number of vertices ($2 \le n \le 1000$), followed by $n - 1$ pairs of vertices u_i, v_i — the edges of the tree (vertices are numbered from 1 to n).

Then the interaction proceeds as follows. Your program makes the first move by printing the yet uncolored vertex number u and the color it wishes to color it c to standard output. Then it must read the opponent's move in the same format, make its move in return, and so on.

When the game is over because your program has won (either by coloring the last vertex, or by forcing the judges' program to color the last vertex), instead of opponent's move your program will read "-1 -1" from standard input. After reading it you program must terminate. Though you can deduce that you have won earlier, or understand that your move is the last move, you should exit only after reading "-1 -1" from standard input.

Examples

standard input	standard output
8 1 2 1 3 2 4 2 5 3 6 3 7 3 8 6 2 8 3 4 2 -1 -1	 1 1 7 2 3 4 2 3

Note

Input is formatted to show which output makes response to the corresponding input. There will be no empty lines in real interaction.

In the given example the judges' program makes the last move (either "5 1", "5 2" or "5 4") but doesn't report it to your program, because after that the game is over and your program has won.