Problem B. Game on Bipartite Graph

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

This is an interactive problem.

You are given a bipartite graph with n vertices in the first part and m vertices in the second part. There are zero or more edges in the graph as well. Multiple edges are allowed. There is also a peg which is initially placed in one of the vertices of the first part. Two players are playing a game on this graph: on each turn, they move the peg along a graph edge. After the peg is moved along an edge, this edge disappears. The player who cannot make a move loses the game. The first player starts the game.

You are given the graph and the initial position of the peg. Your task is to write a program which will play this game as the first player. The jury's program will play as the second player, and it will win the game whenever it is possible. Your program should also win the game whenever it is possible. If it is impossible to win, your program should play until the game ends.

Your opponent's moves will be provided in interactive mode: after each of your moves, you will be given your opponent's next move. Please be sure to use the stream flushing operation after each of your moves to prevent output buffering. For example, you can use "`fflush(stdout)`" in C or C++, "`System.out.flush()`" in Java and "`flush(output)`" in Pascal.

Input

The first line contains four integers n, m, e and v ($1 \le n, m \le 50$, $0 \le e \le 2500$, $1 \le v \le n$): the number of vertices in the first part of the graph, the number of vertices in the second part, the number of edges and 1-based number of vertex in the first part where the peg is placed initially.

Next e lines describe edges. Each of these lines contains two numbers x and y ($1 \le x \le n$, $1 \le y \le m$) separated by a single space, denoting an edge between vertex x of the first part of the graph and vertex y of the second part.

After that, you will be given the opponent's moves. Each move is a 1-based integer v_i ($1 \le v_i \le n$), the number of the vertex in the first part of the graph where your opponent moved the peg.

Output

If you cannot make the next move, just print the phrase "`Player 2 wins`" on a separate line and terminate your solution gracefully. If a move is possible, print a line with a single integer: the number of vertex in the second part of the graph where you move the peg. If after your move, the opponent cannot make a move, additionally print the phrase "`Player 1 wins`" on a new line and terminate your solution gracefully. Do not forget to flush the output buffer after each of your moves.

Each time there are several possible moves, you can pick any one of them, provided that you still win in the end if you initially could.

Examples

standard input	standard output
3 2 5 3 1 1 1 2 2 1 2 2 3 2 1 2	 2 1 2 Player 1 wins
2 2 3 1 1 1 1 2 2 2	 1 Player 1 wins
2 2 6 1 2 2 2 2 1 1 1 2 2 1 2 2 2 1	 1 2 Player 2 wins