

# Problem K

## Reversing Roads


**Problem ID:** reversingroads

**CPU Time limit:** 1 second

**Memory limit:** 1024 MB

**Author:** Greg Hamerly

**Source:** International Collegiate Programming Contest (ACM-I) Dress Rehearsal 2012

**License:** 

You work for the city of One-Direction-Ville. The city mandates that every road in its limits be one direction only. You are evaluating proposals for a new subdivision and its road network. One problem you've observed in some early proposals is that it is impossible to get to certain locations from others along the proposed roads. In order to speed up evaluation of subsequent proposals, you want to write a program to determine if it is possible to get to any location from any other location; you call this a *valid* proposal. And if a proposal is not valid, then your program should find out if there is an easy way to fix it by reversing the direction of one of the roads.

### Input

Input consists of several test cases, at most 5. Each test case begins with a line containing two integers,  $1 \leq m \leq 50$  and  $0 \leq n \leq m(m-1)/2$ .  $m$  indicates the number of locations in the proposal, and  $n$  indicates the number of roads connecting these locations. Following this are  $n$  lines. Each line contains two space-separated integers  $a$  and  $b$ , where  $0 \leq a, b < m$  and  $a \neq b$ . This indicates that there is a road from location  $a$  to location  $b$ . If there is a road from  $a$  to  $b$ , then there will be *no* road from  $b$  to  $a$ . Also, there will never be more than one road between two locations.

The last test case is followed by end-of-file.

### Output

For each case, display the case number followed by an indication of whether the proposal is valid or not. If the proposal is valid, output `valid`. If it is not valid, but by reversing the direction of one roads it can become valid, print the two locations which describe the existing road that should be reversed. If more than one road reversal can create a valid proposal, print the first one that appears in the input. If the proposal is not valid and impossible to become valid by reversing one road, print `invalid`. Follow the format of the sample output.

#### Sample Input 1

```
3 3
0 1
1 2
2 0
3 3
0 1
1 2
0 2
3 2
1 2
0 2
4 4
0 1
1 2
2 3
0 3
```

#### Sample Output 1

```
Case 1: valid
Case 2: 0 2
Case 3: invalid
Case 4: 0 3
```