King Capture

Time limit: *1000 ms*
Memory limit: *128 MB*

</>Template ⇄Interactive

In this problem you should play a game on an undirected graph with n ($1 \leq n \leq 100$) vertices and m ($n - 1 \leq m \leq \binom{n}{2}$) edges. Initially you place a *white* chess king in one of the vertices, then your opponent places a *black* king in another vertex.

Following the initial placement, you and your opponent start performing alternating moves, you being the first to move.

At each turn, the player to move can do one of the following:

- Move her own king (you play with the white king, your opponent with the black one) in one of the vertices adjacent to the current vertex of the king.
- Do nothing

Game goal

You win if at any moment the two kings are in the same vertex.

Your opponent's moves will be performed by a special program called the *interactor*. It is guaranteed that you have a winning strategy on all the given tests. Even more, you want to win the game in as few moves as possible. The interactor will try to maximise the number of moves it takes you to win.

Interaction

First you should read from the standard input two integers n and m.

Each of the next m lines contains 2 integers, representing two vertices that share an edge.

Then you have to print the vertex you initially place the white king. After that you should read the initial position of the black king.

Next, start performing your moves. At each step print the index of the node where you want to move the king, or the index of the current node if you do nothing. Then you should read the index of the node where the interactor moves the black king.

Your program should exit after performing a move that brings the two kings on the same vertex.

Warning: Don't forget to flush after every output operation!

Constraints and notes

- This task is adaptive.
- $2 \leq n \leq 100$
- $n - 1 \leq m \leq \binom{n}{2}$
- The graph is connected, and it doesn't contain multiple edges or self-loops.

Interaction

```
7 11
1 3
1 5
1 6
2 3
2 5
2 6
2 7
3 6
3 7
4 5
5 6
```

```
2
```

```
1
```

```
6
```

```
3
```

```
3
```

```
7 11
1 5
1 6
1 7
2 3
2 4
2 5
3 4
3 5
3 6
4 5
5 6
```

```
1
```

```
2
```

```
5
```

```
3
```

```
3
```