The cop and the robber

In the city of Bytemore the crime level is hitting an all–time high. Among other crimes, roberies are happening every day. If a police officer happens to be nearby, he will try to chase the robber through the small alleys that connect street corners (usually referred to as just *corners*). Unfortunately most of the time robbers escape as they know the district much better than the police.

The Bytemore City Police Department (BCP) is organising a summit in order to reduce the crime rate. One of the initiatives is to use computer aid in pursuing the robbers. For this BCP has made precise maps of all dangerous city districts. Now they need computer software to find best chasing strategies.

The pursuing problem is modelled as follows:

1. A police officer chooses a corner on which to patrol.
2. A robber then chooses a corner for the robbery (robbers always know where police is).
3. The police officer's move consists of him moving to a neighbouring corner (i.e. one that is connected to the current one by an alley) or waiting (i.e. not moving).
4. The robber's move consists of him moving to a neighbouring corner. Note that, unlike the police, robbers cannot wait. It is in their instinct to keep running.
5. The police officer and the robber keep making moves one after another until one of the following happens:
   (a) a situation repeats itself (a situation is defined by police officer's position p, robber's position r and the side T whose turn it is to move next). This corresponds to the robber being able to avoid the police officer indefinitely, so the robber escapes;
   (b) they both meet on the same corner after a move of either of them. In this case the police officer catches the robber.

Task

You have to write a program which, given the plan of a city district, would find whether catching the robber is possible, and if that is the case, would choose the starting position for the police officer and make required moves in order to catch the robber.

Your program must assume that both the police officer and the robber move optimally.

Implementation

You need to implement following two functions:

1. `Initialize(N, M, U[], V[])` — this function takes as parameters number of corners N and number of alleys connecting those corners M. The arrays U and V are each of length M and indicate that there is a street connecting corners $U[i]$ and $V[i]$ for every $0 \leq i \leq M-1$.

   This function has to return `true` if it is possible to catch the robber and `false` otherwise.

2. `GetMove(R)` — this function takes as a parameter the current corner of the robber R and must return the corner of the police officer after his move. When the function is called for the first time, it has to return the starting corner of the police officer.

The function `Initialize` will be called exactly once before any calls to `GetMove`. If it returns `true`, the function `GetMove` will be called zero or more times.

The program will terminate as soon as one of the following happens:

- `GetMove` returns an invalid move.
- A situation repeats itself.
- The robber is caught.

The corners are numbered from 1 to N. You can assume that the robber always makes valid moves, i.e. moves from one street corner to another if there's an alley connecting the two, and never waits.

Example

Provide an example.

Scoring

Explain scoring.

Prepare subtasks.

Subtask 1 (25 points):

Subtask 2 (25 points):

Subtask 3 (25 points):

Subtask 4 (25 points):

Constraints

Check constraints.

Time limit: 1 s.

Memory limit: 256 MB.

Need to add info about graders.