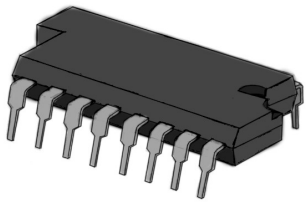
J Juxtaposort

Time limit: 3s

While working on a sorting algorithm, you discover a big problem! The computer chip you are using has a rare but infamous bug, called Broken Arithmetic Primary Comparison: the chip compares floating-point numbers with an offset. To be precise, given two numbers x and y, the comparison checks whether $x - 1 < y$. You want to know whether you can still sort a list of numbers.



The bugged computer chip.

Hidden from you is a sequence of n *distinct* floating-point values $a_1, a_2, \ldots, a_n$. You are given a list of all m pairs (i, j) with $i \neq j$ for which $a_i - 1 < a_j$. Based on this information, can the sequence be unambiguously sorted?

The hidden sequences that the samples are based on, respectively, are $a = [0.9, 1.1]$, $a = [4, 2]$, and $a = [1.7, 1, 2.5]$.

Input

The input consists of:

- One line with two integers n and m ($1 \leq n \leq 10^3, 0 \leq m \leq n(n-1)$), the number of values in the sequence, and the total number of pairs (i, j) for which $a_i - 1 < a_j$.
- m lines, each with two integers i and j ($1 \leq i, j \leq n, i \neq j$), indicating that $a_i - 1 < a_j$. No pair (i, j) is listed more than once.

The m pairs (i, j) are guaranteed to correspond to a sequence of floating-point values.

Output

If the order of the sequence is unambiguously determined, output the indices such that the values are in increasing order. Otherwise, output "`ambiguous`".

Sample Input 1	**Sample Output 1**
2 2 1 2 2 1	ambiguous

Sample Input 2	**Sample Output 2**
2 1 2 1	2 1

Sample Input 3	**Sample Output 3**
3 5 3 1 1 2 2 3 2 1 1 3	2 1 3