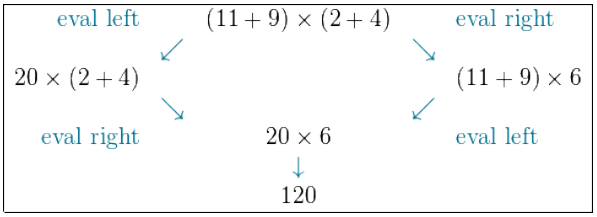
Problem D
Don't Care

Source file name: `dontcare.c`, `dontcare.cpp` *or* `dontcare.java`

In mathematical logic and computer science, an *abstract rewriting system* (ARS) is a convenient framework for describing important properties of artifacts such as logical inference, software systems, social interaction, etc. In its simplest form, an ARS is a set *objects* together with a *binary relation* on the objects expressing how these can transition.

Some ARS are of very special interest because even if an object can make a transition in more than one way, these transitions will eventually yield the same result. Such an ARS guarantees that if some irreducible object is reached by successive transition steps, then this is also the case independently of the choice of the first transition taken. In other words, an ARS with this property converts nondeterminism from "don't know" into "don't care", thus avoiding the need for backtracking. Let us agree to call an ARS with this property a *Don't Care ARS*.

Consider the usual rules of elementary arithmetic: they form an abstract rewriting system. For example, the expression $(11+9)\times(2+4)$ can be evaluated starting either at the left or at the right parentheses; however, in both cases the same *irreducible* result is obtained eventually. This suggests that the arithmetic reduction system is don't care.



Borrowed from Wikipedia.org

There is no general purpose algorithm that can *always* determine if a given ARS is don't care, i.e., checking the don't care property is in general undecidable for arbitrary ARS. However, because of its importance, this property has been thoroughly studied and decision procedures have been obtained for restricted versions of the problem. In particular, the don't care property can always be checked mechanically for systems having a finite number of objects.

Let $\mathcal{A}=(A,\rightarrow)$ denote an ARS with a set of objects A and a transition relation $\rightarrow$ on A (that is, $\rightarrow\subseteq A^2$). Then, $\mathcal{A}$ is don't care if and only if the following two conditions are satisfied:

1. the relation $\rightarrow$ is well-founded, i.e., no object can be reduced with $\rightarrow$ indefinitely, and
2. for any $a, b, c\in A$, if there are direct $\rightarrow$-transitions from a to b and from a to c, then there is $d\in A$ such that d is reachable via (zero or more) $\rightarrow$-transitions both from b and c.

You have been assigned the task of implementing an efficient algorithm for deciding the don't care property for finite ARS.

Input

The input consists of several test cases. The first line of each test case contains two numbers n and m ($1 \leq n \leq 1000$, $0 \leq m \leq 30000$) defining, respectively, the number of objects and the number of transitions in the ARS. Each of the next m lines contains a pair of blank-separated integers a, b indicating that there is a transition in the ARS from object a to object b ($0 \leq a < n$, $0 \leq b < n$). The last test case is followed by a line with two zeros "`0 0`".

The input must be read from standard input.

Output

For each ARS output exactly one line. If the input ARS is don't care, then output '`1`'; otherwise, output '`0`'.

The output must be written to standard output.

Sample input	**Output for the sample input**
3 2 0 1 1 2 2 2 0 1 0 1 2 2 0 1 1 0 3 2 0 1 0 2 0 0	1 1 0 0