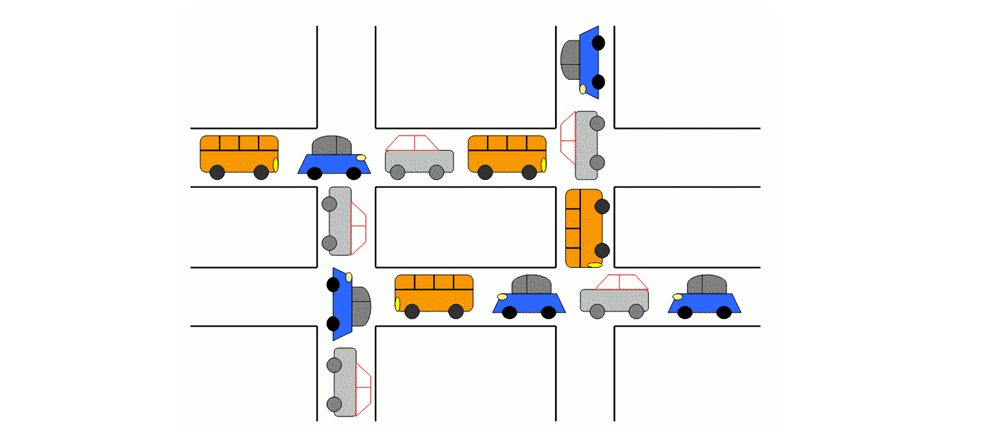
**WARNING! Deadlock!**

Time Limit: 1000ms

Memory Limit: 64 MB

“Ah... what a beautiful day is today, just like every other day!”, said Arataki Itto who is working as a software engineer at Booble. But it’s about to change pretty soon.



One day, he was deeply puzzled by an unusual behavior in the distributed system for which he was responsible for. He had observed that the system had become significantly slower than usual, and he was determined to investigate this issue further. Initially, he looked for the potential causes of this problem and suggested that it might be related to a type of deadlock known as circular wait. Since the number of processes is quite high, he had an idea to create a program that would help him determine whether a circular wait deadlock was occurring in the system. Unfortunately, he was very busy and he asked his junior to do this instead.

As Arataki’s Itto junior, you wanted to help him to solve this problem. You then remembered that circular wait is a condition involving three or more processes, where one process is waiting for the resource, which is held by the second process, which is also waiting for the resource held by the third process, etc. This will continue until the last process, which is waiting for a resource held by the first process. This creates a circular chain. Simply put, one process is waiting for another process in a circular manner to finish and release the resources it is using.

There are N processes currently running in the system and M descriptions about them. Each of the M description, denoted as a pair of u and v, describes that process u is waiting for process v to finish and release its resource (but not the other way around). Below is the input format that your program has to be able to process.

**INPUT**

* The first line of the input contains two integers separated by a space, N (3 ≤ N ≤ 105), the number of processes, followed by M (1 ≤ M ≤ min(, 105)), the number of descriptions to the processes.
* The next M lines each contains a pair of integers separated by a space, u and v (1 ≤ u, v ≤ N), that denotes process u is waiting for process v.

**OUTPUT**

* Output one line that contains one of the following:
* **“[WARNING] Deadlock Detected!”** without double quotes, if there is at least one circular wait deadlock detected by the program.
* **“[INFO] Everything OK”** without double quotes, if there is no circular wait deadlock detected by the program.

**SAMPLE TEST CASE 1**

|  |  |
| --- | --- |
| **INPUT** | **OUTPUT** |
| 3 3  1 2  2 3  3 1 | [WARNING] Deadlock Detected! |

**SAMPLE TEST CASE 2**

|  |  |
| --- | --- |
| **INPUT** | **OUTPUT** |
| 6 5  2 5  5 6  5 1  1 4  3 5 | [INFO] Everything OK |

**SAMPLE TEST CASE 3**

|  |  |
| --- | --- |
| **INPUT** | **OUTPUT** |
| 9 8  8 7  7 9  1 5  5 2  2 4  4 3  3 6  6 2 | [WARNING] Deadlock Detected! |