Problem B. Blocking Buffer

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

New microarchitecture for parallel programming has a pipe buffer to organize the data exchange between two execution threads. The buffer is circular and can contain no more than l bytes. At the beginning of the program execution the buffer is empty. The first thread sometimes tries to write to the buffer, and it always uses blocks of size w bytes. The second thread sometimes reads from the buffer in blocks of size r bytes each. All write operations put the new data at the end of the pipe buffer and all read operations take the data from the beginning of the pipe buffer, so empty positions always form one continuous segment.

Both write and read operations are blocking. It means that, if the first thread tries to write the block of data to the buffer but there is not enough space there, the thread stops and waits until there will be at least w empty bytes in the buffer. Similarly, if the second thread tries to read the block of data from the buffer but there are less than r bytes of data left, the thread stops and waits until there will be enough new data in the buffer.

In this particular problem, a *deadlock* is a situation when the first thread is blocked because there is not enough space in the buffer to fit a new block of size w, and the second thread is blocked because there is not enough data in the buffer to read a new block of size r.

Your goal is to determine whether there exists such a sequence of reads and writes from two threads that will result in a deadlock. Note that write operations are only performed by the first thread and read operations are only performed by the second thread.

Input

The only line of input contains three integers l, r and w ($0 < l, r, w \le 10^{18}$, $r \le l$, $w \le l$).

Output

If there exists a sequence of reads and writes that result in a deadlock, print "`DEADLOCK`" (without quotes) in the only line of the output. Otherwise, print "`OK`" (without quotes).

Examples

standard input	standard output
5 3 4	DEADLOCK
5 2 3	OK

Note

One of the ways to obtain the deadlock in the first sample is:

1. The first thread writes a block of size 4 to the buffer.
2. The second thread reads a block of size 3. There is 1 byte of data left in the buffer.
3. The first thread writes a block of size 4. The buffer is full now.
4. The second thread reads a block of size 3. There are 2 bytes of data left in the buffer.
5. The second thread tries to read a block of size 3, but there is not enough data in the buffer, so the thread is blocked.
6. The first thread tries to write a block of size 4, but there it not enough free space in the buffer, so the thread is also blocked. The deadlock just happened.