XOR

The operation “bitwise exclusive or” (we denote it with ⊕) is **standardly** defined on each couple of non-negative integers (*a*, *b*) as follows:

Let and be the *n*-digit binary notations of the numbers *a* and *b*, i. e., *a*i and *b*i are zeroes or ones (if the binary digits of the smaller one are less than *n*, its notation is filled up with “leading zeroes”). Then the number *c* = *a* ⊕ *b* is defined in this way: its *i*th binary digit *c*i () is obtained by applying the operation “exclusive or” on the *i*th binary digits of *a* and *b* respectively, i. e., *ci* = *ai* xor *bi* for each *i* from 0 to *n*-1. The xor operation is defined on binary digits as follows: 0 xor 0 = 0; 0 xor 1 = 1; 1 xor 0 = 1; 1 xor 1 = 0.

The operation is easily extended for more operands. More specifically, for the consecutive positive integers in the interval [*a*, *b*] we can denote  = *а*⊕(*а*+1)⊕(*a*+2)⊕…⊕*b*, assuming operation execution from left to right. Consider the positive integers *a* and *b* (*a*<*b*), defining the closed interval of integers [*a*, *b*], as well as the positive integer *n* (1<*n*≤*b*‑*a*+1). Consider the operation “bitwise exclusive or” on every possible *n*-tuple of consecutive integers in the interval [*a*, *b*].

Write a program **xor** to find out the largest value *M* which this process can produce.

Let’s, for clarity, take a closer look at the case *a*=10, *b*=20, *n*=6. I. e., we consider the interval [10, 20] of integers, more precisely – all sextuples of consecutive integers in it. For each of them we apply the generalized operation “bitwise exclusive or”:

10⊕11⊕12⊕13⊕14⊕15=10102⊕10112⊕11002⊕11012⊕11102⊕11112=00012=1;

11⊕12⊕13⊕14⊕15⊕16=010112⊕011002⊕011012⊕011102⊕011112⊕100002=110112=27;

12⊕13⊕14⊕15⊕16⊕17=011002⊕011012⊕011102⊕011112⊕100002⊕100012=000012=1;

13⊕14⊕15⊕16⊕17⊕18=011012⊕011102⊕011112⊕100002⊕100012⊕100102=111112=31;

14⊕15⊕16⊕17⊕18⊕19=011102⊕011112⊕100002⊕100012⊕100102⊕100112=000012=1;

15⊕16⊕17⊕18⊕19⊕20=011112⊕100002⊕100012⊕100102⊕100112⊕101002=110112=27.

Obviously, in this case the solution is 31, resulting in the sextuple which starts with 13.

Input

One line is read from the standard input, containing the space separated positive integers *a*, *b* and *n*.

Output

The program should write to the standard output one line, containing only one non-negative integer *M* which is the biggest possible number, obtained by applying the operation “bitwise exclusive or” on at least one of the *n*-tuples of consecutive integers in the interval [*a*, *b*].

Constraints

*a*, *b* and *n* are positive integers with no more than 18 decimal digits; *a* < *b*; 1 < *n* ≤ *b* – *a*+ 1.

Subtask 1

In 20% of the cases *a*, *b* and *n* do not exceed 107.

Subtask 2

In other 20% of the cases holds *n* < 5.107.

Subtask 3

In other 20% of the cases *n* is odd for sure.

Subtask 4

In the last 40% of the cases holds *n* < 108.

Example

Input

10 20 6

Output

31