INTERNATIONAL TOURNAMENT IN INFORMATICS
Shumen, 23 November 2013, Senior Group

TASK A2. 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 $a = \overline{a_{n-1}a_{n-2}a_{n-3}\cdots a_0}$ and $b = \overline{b_{n-1}b_{n-2}b_{n-3}\cdots b_0}$ 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 $c = a \oplus b$ is defined in this way: its i^{th} binary digit c_i ($c = \overline{c_{n-1}c_{n-2}c_{n-3}\cdots c_0}$) is obtained by applying the operation "exclusive or" on the i^{th} binary digits of *a* and *b* respectively, i. e., $c_i = a_i$ xor b_i 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 $\oplus_{i=a}^{b} i = a\oplus(a+1)\oplus(a+2)\oplus\ldots\oplus b$, 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* ($1<n\le b-a+1$). 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":

$10\oplus11\oplus12\oplus13\oplus14\oplus15=1010_2\oplus1011_2\oplus1100_2\oplus1101_2\oplus1110_2\oplus1111_2=0001_2=1$;
$11\oplus12\oplus13\oplus14\oplus15\oplus16=01011_2\oplus01100_2\oplus01101_2\oplus01110_2\oplus01111_2\oplus10000_2=11011_2=27$;
$12\oplus13\oplus14\oplus15\oplus16\oplus17=01100_2\oplus01101_2\oplus01110_2\oplus01111_2\oplus10000_2\oplus10001_2=00001_2=1$;
$13\oplus14\oplus15\oplus16\oplus17\oplus18=01101_2\oplus01110_2\oplus01111_2\oplus10000_2\oplus10001_2\oplus10010_2=11111_2=31$;
$14\oplus15\oplus16\oplus17\oplus18\oplus19=01110_2\oplus01111_2\oplus10000_2\oplus10001_2\oplus10010_2\oplus10011_2=00001_2=1$;
$15\oplus16\oplus17\oplus18\oplus19\oplus20=01111_2\oplus10000_2\oplus10001_2\oplus10010_2\oplus10011_2\oplus10100_2=11011_2=27$.

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; $a < b$; $1 < n \le b - a + 1$.

- In 20% of the cases *a*, *b* and *n* do not exceed 10^7.
- In other 20% of the cases holds $n < 5.10^7$.
- In other 20% of the cases *n* is odd for sure.
- In the last 40% of the cases holds $n < 10^8$.

Example

Input
```
10 20 6
```

Output
```
31
```