**INTERNATIONAL TOURNAMENT IN INFORMATICS**

**Shumen, 23 November 2013, Senior Group**

**TASK A2. XOR**

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

Let $a = \overline{a_{n-1}a_{n-2}a_{n-3} \ldots a_0}$ and $b = \overline{b_{n-1}b_{n-2}b_{n-3} \ldots b_0}$ 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 $c_i$ ($c = c_{n-1}c_{n-2}c_{n-3} \ldots c_0$) is obtained by applying the operation “exclusive or” on the $i^{th}$ binary digits of $a$ and $b$ respectively, i. e., $c_i = a_i \oplus b_i$ for each $i$ from 0 to $n-1$. The xor operation is defined on binary digits as follows: $0 \oplus 0 = 0$; $0 \oplus 1 = 1$; $1 \oplus 0 = 1$; $1 \oplus 1 = 0$.

The operation is easily extended for more operands. More specifically, for the consecutive positive integers in the interval $[a, b]$ we can denote $\oplus_{i=a}^{b} = a \oplus (a+1) \oplus (a+2) \ldots \oplus b$, 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 $n$ ($1 < n \leq b - a + 1$). Consider the operation “bitwise exclusive or” on every possible $n$-tuples 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”:

\[
\begin{align*}
10 \oplus 11 & \oplus 12 \oplus 13 \oplus 14 \oplus 15 = 1010_{2} \oplus 1011_{2} \oplus 1100_{2} \oplus 1101_{2} \oplus 1110_{2} \oplus 1111_{2} = 0001_{2} = 1; \\
11 \oplus 12 & \oplus 13 \oplus 14 \oplus 15 \oplus 16 = 0101_{2} \oplus 0110_{2} \oplus 0111_{2} \oplus 0111_{2} \oplus 1000_{2} \oplus 1000_{2} = 1101_{2} = 27; \\
12 \oplus 13 & \oplus 14 \oplus 15 \oplus 16 \oplus 17 = 0110_{2} \oplus 0111_{2} \oplus 0110_{2} \oplus 0111_{2} \oplus 1000_{2} \oplus 1001_{2} = 0000_{2} = 1; \\
13 \oplus 14 & \oplus 15 \oplus 16 \oplus 17 \oplus 18 = 0111_{2} \oplus 0110_{2} \oplus 0111_{2} \oplus 1000_{2} \oplus 1001_{2} \oplus 1001_{2} = 1111_{2} = 31; \\
14 \oplus 15 & \oplus 16 \oplus 17 \oplus 18 \oplus 19 = 0111_{2} \oplus 0111_{2} \oplus 1000_{2} \oplus 1001_{2} \oplus 1010_{2} \oplus 1011_{2} = 0000_{2} = 1; \\
15 \oplus 16 & \oplus 17 \oplus 18 \oplus 19 \oplus 20 = 1000_{2} \oplus 1000_{2} \oplus 1001_{2} \oplus 1010_{2} \oplus 1011_{2} \oplus 1100_{2} = 1101_{2} = 27.
\end{align*}
\]

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

- In 20% of the cases $a$, $b$ and $n$ do not exceed $10^7$.
- In other 20% of the cases holds $n < 5 \cdot 10^7$.
- 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
```