3133. Minimum Array End

You are given two integers `n` and `x`. You have to construct an array of **positive** integers `nums` of size `n` where for every `0 <= i < n - 1`, `nums[i + 1]` is **greater than** `nums[i]`, and the result of the bitwise `AND` operation between all elements of `nums` is `x`.

Return the **minimum** possible value of `nums[n - 1]`.

Example 1:

Input: n = 3, x = 4

Output: 6

Explanation:

`nums` can be `[4,5,6]` and its last element is 6.

Example 2:

Input: n = 2, x = 7

Output: 15

Explanation:

`nums` can be `[7,15]` and its last element is 15.

Constraints:

- `1 <= n, x <= 10^8`

3133. Minimum Array End

```
/*
   Consecutive ORing
   Time complexity: O(n)
   Space complexity=o(1)
*/
class Solution {
   public:
      long long minEnd(int n, int x){
         // First number is equal to x
         long long y=x;

         // For the next n-1 numbers: number are incresing
         // without changing the set bits.
         for(int i=1;i<n;++i) y=(y+1)|x;

         return y;
      }
};
```

3133. Minimum Array End

```
/*
   Bits interleaving
   Time complexity: O(64)=O(1)
   Space complexity=o(1)
*/
typedef long long ll;
class Solution {
   public:
      ll minEnd(int n, int x){
         // Need the n-1 to interleave with x
         n--;

         // First number is equal to x
         // change x to long long
         ll y=x;

         int i=0; // run over x
         int j=0; // run over n-1
         while(i<64){
            // In x: If the bit is set don't change it
            if(y&(1ll<<i)){
               i++; // Go next bit
               continue;
            }
            // if the i-th bit is not set in x, change it with the j-th set bit in n-1
            if(n&(1ll<<j)) y|=(1ll<<i);

            i++;
            j++;
         }
         return y;
      }
};
```