[A - Lowest Bit](https://vjudge.net/problem/EOlymp-1753" \t "_blank)

Given an positive integer **A** (**1** ≤ **A**≤**100**), output the lowest bit of **A**.

For example, given **A = 26**, we can write A in binary form as **11010**, so the lowest bit of **A** is **10**, so the output should be **2**.

Another example goes like this: given **A = 88**, we can write **A** in binary form as **1011000**, so the lowest bit of **A** is **1000**, so the output should be **8**.

**Input**

Each line of input contains only an integer **A** (**1** ≤ **A**≤**100**). A line containing "**0**" indicates the end of input, and this line is not a part of the input data.

**Output**

For each **A** in the input, output a line containing only its lowest bit.

**Example 1**

Input example #1

26

88

0

Output example #1

2

8

Lowest Bit

We need to find the first set bit position going from RIGHT to LEFT in the number.  and 1 << bitPosition will be the answer.

To find right most set bit we can use following approaches

Loop with Right Shifting unless we find a 1

long long lowestBit(long long n){  
    long long p = 0;  
    while(n!=0)  
        if(n&1)  
            break;  
        n >>= 1;  
        p++;  
    return 1 << p;  
}

One Liner Solution using (n & (n-1))

FACT: n = (n & (n-1)) // RESETS RIGHT MOST SET BIT OF N  
  
So if  
n = 0010 0110  
n & (n-1) = 0010 0100  
-----------------------  
If we XOR above two statements, we will be left with desired answer.

long long lowestBit(long long n){  
    return (n ^ (n & (n-1)));  
}

One Liner Solution using (n & -n)

FACT: n = (n & -n) // GIVES BINARY PATTERN WITH ONLY RIGHT MOST BIT OF n SET

x = n & -n  
  
 n = 0010 0110  
-n = 1101 1010  
----------------  
 x = 0000 0010  
 ----------------

long long lowestBit(long long n){  
    return (n & -n);  
}

#include <iostream>

using namespace *std*;

int main(void) {

int n; *cin* >> n;

while (n != 0)

{

int count = 0;

for (auto i = 0; i < 32; i++)

{

bool lastBit = n & 1;

if (!lastBit)

count++;

else

break;

n = n >> 1;

}

*cout* << (1 << count) << *endl*;

*cin* >> n;

}

return 0;

}