In order to stop the Mad Coder evil genius you need to decipher the encrypted message he sent to his minions. The message contains several numbers that, when typed into a supercomputer, will launch a missile into the sky blocking out the sun, and making all the people on Earth grumpy and sad.

You figured out that some numbers have a modified single digit in their binary representation. More specifically, in the given number n the kth bit from the right was initially set to 0, but its current value might be different. It's now up to you to write a function that will change the kth bit of n back to 0.

**Example**

* For n = 37 and k = 3, the output should be  
  killKthBit(n, k) = 33.

3710 = 100**1**012 ~> 100**0**012 = 3310.

* For n = 37 and k = 4, the output should be  
    
  killKthBit(n, k) = 37.

The 4th bit is 0 already (looks like the Mad Coder forgot to encrypt this number), so the answer is still 37.

**Input/Output**

* **[time limit] 3000ms (cs)**
* **[input] integer n**

*Constraints:*  
0 ≤ n ≤ 231 - 1.

* **[input] integer k**

The 1-based index of the changed bit (counting from the right).

*Constraints:*  
1 ≤ k ≤ 31.

* **[output] integer**

<https://codefights.com/arcade/code-arcade/corner-of-0s-and-1s/b5z4P2r2CGCtf8HCR>

int killKthBit(int n, int k)

{

return n &= ~(1 << (k - 1));

}

<http://stackoverflow.com/questions/1873505/bit-operations-in-c>

|  |  |
| --- | --- |
| 4down vote | In general, counting the least significant bit as 0, to set bit N, you need to OR the original value with 1 << N.  Eg to set bit 1:  val |= (1 << 1);  To clear bit N, you need to AND the original value with the bit-wise inverse of 1 << N.  Eg to clear bit 1:  val &= ~(1 << 1); |