You're given an arbitrary 32-bit integer n. Swap each pair of adjacent bits in its binary representation and return the result as a decimal number.

**Example**

* For n = 13, the output should be  
  swapAdjacentBits(n) = 14.

1310 = 11012 ~> 11102 = 1410.

* For n = 74, the output should be  
  swapAdjacentBits(n) = 133.

7410 = 010010102 ~> 100001012 = 13310.  
Note the preceding zero written in front of the initial number: since both numbers are 32-bit integers, they have 32 bits in their binary representation. The preceding zeros in other cases don't matter, so they are omitted. Here, however, it does make a difference.

**Input/Output**

* **[time limit] 20000ms (swift)**
* **[input] integer n**

*Constraints:*  
0 ≤ n < 230.

* **[output] integer**

<https://codefights.com/arcade/code-arcade/corner-of-0s-and-1s/dShYZZT4WmvpmfpgB>

<https://github.com/Swift-Solutions/CodeFights/blob/master/solutions/swap_adjacent_bits.swift>

|  |
| --- |
| func swapAdjacentBits(n: Int) -> Int { |
|  | return { (n: Int) -> Int in |
|  | var n = n, res = 0, base = 1 |
|  | while n > 0 { |
|  | var a = n % 2 |
|  | n /= 2 |
|  | var b = n % 2 |
|  | n /= 2 |
|  | res += (a \* 2 + b) \* base |
|  | base \*= 4 |
|  | } |
|  | return res |
|  | }(n) ; |
|  | } |