Given an integer value n, you need to find whether the binary representation of n is a special palindrome or not.

A number (in binary format) is said to be a special palindrome if:

* It's a palindrome;
* It contains exactly one 0;
* It contains at least one 1.

**Example**

* For n = 5, the output should be  
  isSpecialPalindrome(n) = true.  
  As 510 = 1012, and 101 is a palindrome, contains exactly one 0 and contains at least one 1 (here are two 1-s), so it's a special palindrome.
* For n = 0, the output should be  
  isSpecialPalindrome(n) = false.  
  As 010 = 02, and although 0 is a palindrome and contains only one 0 in it, but since it does not contain any 1-s, it is not a special palindrome.
* For n = 7, the output should be  
  isSpecialPalindrome(n) = false.  
  As 710 = 1112, and although 111 is a palindrome and contains more than one 1-s in it, but since it does not contain one 0, it is not a special palindrome.

**Input/Output**

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

An integer in base 10.

*Guaranteed constraints:*  
0 ≤ n ≤ 109.

* **[output] boolean**

Return true if the binary representation of the n is a special palindrome, falseotherwise.

<https://codefights.com/challenge/H3xfuW74xaSgAXaSt/solutions>

static bool isSpecialPalindrome(int n)

{

char[] bin = Convert.ToString(n, 2).ToCharArray();

string copia = new string(bin);

Array.Reverse(bin);

if (new string(bin) == copia && bin.Count(e => e == '0') == 1 && bin.Count(e => e == '1') > 1)

{

return true;

}

return false;

}