You're given two integers, n and m. Find position of the rightmost pair of equal bits in their binary representations (it is guaranteed that such a pair exists), counting from right to left.

Return the value of 2position\_of\_the\_found\_pair (0-based).

**Example**

For n = 10 and m = 11, the output should be  
equalPairOfBits(n, m) = 2.

1010 = 10**1**02, 1110 = 10**1**12, the position of the rightmost pair of equal bits is the bit at position1 (0-based) from the right in the binary representations.  
So the answer is 21 = 2.

**Input/Output**

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

*Constraints:*  
0 ≤ n ≤ 230.

* **[input] integer m**

*Constraints:*  
0 ≤ m ≤ 230.

* **[output] integer**

<https://codefights.com/challenge/ATLKot3MTscKu2iqq/main?utm_source=featuredChallenge&utm_medium=email&utm_campaign=email_notification>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static string toBin(int n)

{

string bin = "";

while (n > 0)

{

bin = (n % 2).ToString() + bin;

n /= 2;

}

return bin;

}

static int equalPairOfBits(int n, int m)

{

string binN = toBin(n);

string binM = toBin(m);

int i = binN.Length - 1;

int j = binM.Length - 1;

int pos = 0;

while (i >= 0 && j >= 0)

{

if (binN[i] == binM[j])

{

//pos = binN.Length - i+1;

break;

}

pos++;

i--;

j--;

}

return (int)Math.Pow(2, pos);

}

static void Main(string[] args)

{

//Console.WriteLine(toBin(2));

Console.WriteLine(equalPairOfBits(28,27));

Console.ReadLine();

}

}

}