Consider a number m = 2p - 1, where p is prime. If m is itself prime it is known as a Mersenne prime.

Given p determine whether this exponent yields a Mersenne prime.

**Example**

* For p = 3, the output should be  
  mersenne(p) = true.  
  m = 23 - 1 = 7, and 7 is prime.
* For p = 11, the output should be  
  mersenne(p) = false.  
   = 211 - 1 = 2047, and 2047 is NOT prime.

**Input/Output**

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

A prime number.

*Guaranteed constraints:*  
2 < p < 1500.

* **[output] boolean**

true if 2p - 1 is prime, false otherwise.

<https://codefights.com/challenge/EBGBqS5pbyBbHATgm/solutions>

Lista de primos Mersenne:

<https://www.mersenne.org/primes/>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static bool mersenne(int p)

{

int[] pot = {2,3,5,7,13,17,19,31,61,89,107,127,521,607,1279,2203};

return pot.Contains(p);

}

static void Main(string[] args)

{

Console.WriteLine(mersenne(53));

Console.ReadLine();

}

}

}

bool mersenne(int p)

{

var s = new System.Numerics.BigInteger(4);

var M = System.Numerics.BigInteger.Pow(2, p) - 1;

while (--p > 1)

s = (s \* s - 2) % M;

return s == 0;

}

bool mersenne(int p)

{

System.Numerics.BigInteger m = System.Numerics.BigInteger.Pow(2, p) - 1;

System.Numerics.BigInteger s = 4;

//while (s <= 0)

//{

for (int x = 0; x < p - 2; x++)

{

s = ((s \* s) - 2) % (System.Numerics.BigInteger)m;

if (s == 0)

break;

}

return s == 0;

}