Consider the following operation - we take a positive integer n and replace it with the sum of its [prime](keyword://prime-number) factors (if a prime number is presented multiple times in the factorization of n, then it's counted the same number of times in the sum). This operation is applied sequentially first to the given number, then to the first result, then to the second result and so on, until the result remains the same.

Given any number, find the final result of the operation.

**Example**

For n = 24, the output should be  
factorSum(n) = 5.

24 -> (2 + 2 + 2 + 3) = 9 -> (3 + 3) = 6 -> (2 + 3) = 5 -> 5.  
So the answer for n = 24 is 5.

**Input/Output**

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

*Guaranteed constraints:*  
2 ≤ n ≤ 200.

* **[output] integer**

<https://codefights.com/challenge/7WDDw8v7QKHaHN6bK/solutions>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static List<int> G(int n)

{

List<int> p = new List<int>();

for (int d = 2; d <= n; d++)

{

while (n % d == 0)

{

p.Add(d);

n = n / d;

}

}

return p;

}

static bool esPrimo(int n)

{

if (n < 2) return false;

if (n == 2) return true;

if (n % 2 == 0) return false;

int sqr = (int)Math.Sqrt(n);

for (int i = 3; i <= sqr; i += 2)

{

if (n % i == 0) return false;

}

return true;

}

static List<int> primeFactors(int n)

{

List<int> factors = new List<int>();

int divisor = 2;

while (n >= 2)

{

if (n % divisor == 0)

{

factors.Add(divisor);

n /= divisor;

}

else

{

divisor++;

}

}

return factors;

}

static int factorSum(int n)

{

List<int> f = primeFactors(n);

while (f.Count != 1)

{

f = primeFactors(f.Sum());

//if (esPrimo(f.Sum())) break;

if (f.Sum() == 4) return 4;

}

return f.Sum();

}

static void Main(string[] args)

{

Console.WriteLine( factorSum(4));

Console.ReadLine();

}

}

}