We define the *weakness* of number x as the number of positive integers smaller than x that have more divisors than x.

It follows that the *weaker* the number, the greater overall *weakness* it has. For the given integer n, you need to answer two questions:

* what is the *weakness* of the *weakest*numbers in the range [1, n]?
* how many numbers in the range [1, n]have this *weakness*?

Return the answer as an array of two elements, where the first element is the answer to the first question, and the second element is the answer to the second question.

**Example**

For n = 9, the output should be  
weakNumbers(n) = [2, 2].

Here are the number of divisors and the specific *weakness* of each number in range [1, 9]:

* 1: d(1) = 1, weakness(1) = 0;
* 2: d(2) = 2, weakness(2) = 0;
* 3: d(3) = 2, weakness(3) = 0;
* 4: d(4) = 3, weakness(4) = 0;
* 5: d(5) = 2, weakness(5) = 1;
* 6: d(6) = 4, weakness(6) = 0;
* 7: d(7) = 2, weakness(7) = 2;
* 8: d(8) = 4, weakness(8) = 0;
* 9: d(9) = 3, weakness(9) = 2.

As you can see, the maximal *weakness* is 2, and there are 2 numbers with that *weakness* level.

**Input/Output**

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

*Constraints:*  
1 ≤ n ≤ 1000.

* **[output] array.integer**

Array of two elements: the *weakness* of the *weakest* number, and the number of integers in range [1, n] with this *weakness*.

<https://codefights.com/arcade/code-arcade/labyrinth-of-nested-loops/oL7YuygJKtMSNLeJn>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static int cantDivisores(int n)

{

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

int cantDiv = 0;

for (int i = 1; i <= sqr; i++)

{

if (n % i == 0)

{

cantDiv++;

}

}

cantDiv \*= 2;

if (sqr \* sqr == n)

{

cantDiv--;

}

return cantDiv;

}

static int debilidad(int x)

{

int divx = cantDivisores(x);

int debil = 0;

for (int i = 1; i < x; i++)

{

if (cantDivisores(i) > divx)

{

debil++;

}

}

return debil;

}

static int[] weakNumbers(int n)

{

int maxDebilidad = 0, cantidadDebiles =0;

for (int i = 1; i <= n; i++)

{

//maxDebilidad = Math.Max(maxDebilidad, debilidad(i));

if (debilidad(i) > maxDebilidad)

{

maxDebilidad = debilidad(i);

cantidadDebiles = 1;

}

else if (debilidad(i) == maxDebilidad)

{

cantidadDebiles++;

}

}

int[] ans = new int[2];

ans[0] = maxDebilidad;

ans[1] = cantidadDebiles;

//Console.WriteLine(maxDebilidad + " " + cantidadDebiles);

return ans;

}

static void Main(string[] args)

{

// Console.WriteLine(cantDivisores(12));

Console.WriteLine(weakNumbers(500));

Console.ReadLine();

}

}

}