**5 kyu**

**Integers: Recreation One**

6389188% of 1,063612 of6,626[g964](https://www.codewars.com/users/g964)

C#

* [TRAIN AGAIN](https://www.codewars.com/kata/integers-recreation-one/train/csharp)
* [NEXT KATA](https://www.codewars.com/trainer/csharp)

Details

[Solutions](https://www.codewars.com/kata/integers-recreation-one/solutions/csharp)

[Forks (5)](https://www.codewars.com/kata/integers-recreation-one/forks/csharp)

[Discourse (169)](https://www.codewars.com/kata/integers-recreation-one/discuss/csharp)

* Add to Collection
* |
* Share this kata:

Divisors of 42 are : 1, 2, 3, 6, 7, 14, 21, 42. These divisors squared are: 1, 4, 9, 36, 49, 196, 441, 1764. The sum of the squared divisors is 2500 which is 50 \* 50, a square!

Given two integers m, n (1 <= m <= n) we want to find all integers between m and n whose sum of squared divisors is itself a square. 42 is such a number.

The result will be an array of arrays or of tuples (in C an array of Pair) or a string, each subarray having two elements, first the number whose squared divisors is a square and then the sum of the squared divisors.

#Examples:

list\_squared(1, 250) --> [[1, 1], [42, 2500], [246, 84100]]

list\_squared(42, 250) --> [[42, 2500], [246, 84100]]

The form of the examples may change according to the language, see Example Tests: for more details.

**Note**

In Fortran - as in any other language - the returned string is not permitted to contain any redundant trailing whitespace: you can use dynamically allocated character strings.

<https://www.codewars.com/kata/integers-recreation-one/csharp>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

//static int chessKnightMoves(string cell)

//{

//}

static long SumaDivisores(long n)

{

//List<int> div = new List<int>();

long sum = 0;

for (long i = 1; i \* i <= n; i++)

{

if (n % i == 0)

{

// div.Add(i);

sum += i \* i;

if (n / i != i)

{

// div.Add(n / i);

sum += (n / i) \* (n / i);

}

}

}

//return div;

return sum;

}

public static string listSquared(long m, long n)

{

//string ans = "";

List<long[]> ans = new List<long[]>();

// your code

for (long i = m; i <= n; i++)

{

long suma = SumaDivisores(i);

long sqr = (long)Math.Sqrt(suma);

if (sqr \* sqr == suma)

{

long[] par = new long[2];

par[0] = i;

par[1] = suma;

ans.Add(par);

}

}

string concat = "[";

for (int i = 0; i < ans.Count; i++)

{

concat += "[" + ans[i][0] + ", " + ans[i][1] + "], ";

}

concat = concat.TrimEnd(' ');

concat = concat.TrimEnd(',');

concat += "]";

return concat;

}

static void Main(string[] args)

{

//divisores(42);

// Console.WriteLine(listSquared(250, 500));

Console.WriteLine(listSquared(42, 250));

Console.ReadLine();

}

}

}

* **using System;**
* **using System.Collections.Generic;**
* **using System.Linq;**
* **public class SumSquaredDivisors**
* **{**
* **public static string listSquared(long m, long n)**
* **{**
* **List<string> result = new List<string>();**
* **for(int i = (int)m; i <= n; i++)**
* **{**
* **int sum = GetDivisors(i).Select(x => x\*x).Sum();**
* **if(IsSquare(sum))**
* **{**
* **result.Add(string.Format("[{0}, {1}]", i, sum));**
* **}**
* **}**
* **return "[" + string.Join(", ", result) + "]";**
* **}**
* **private static bool IsSquare(int num)**
* **{**
* **return Math.Sqrt(num) % 1 == 0;**
* **}**
* **private static List<int> GetDivisors(int num)**
* **{**
* **List<int> divs = new List<int>();**
* **for(int i = 1; i <= num; i++)**
* **{**
* **if(num % i == 0)**
* **divs.Add(i);**
* **}**
* **return divs;**
* **}**
* **}**
* 2 similar code variations are grouped with this oneShow Variations
  + - Best Practices11
    - Clever1
  + 2
  + [Fork](https://www.codewars.com/kumite/new?group_id=56c6d8bdd3702b1219001b58&review_id=55aa15eec1eba8a65a000140)
  + Compare with your solution
  + [Link](https://www.codewars.com/kata/reviews/55aa15eec1eba8a65a000140/groups/56c6d8bdd3702b1219001b58)
* [BPCJr](https://www.codewars.com/users/BPCJr)
* **using System;**
* **public class SumSquaredDivisors**
* **{**
* **public static string listSquared(long m, long n)**
* **{**
* **string output = "[";**
* **for (long i = m; i <= n; i++)**
* **{**
* **long sum = 0;**
* **for (long j = 1; j <= Math.Sqrt(i); j++)**
* **{**
* **if (i % j == 0)**
* **{**
* **sum += j \* j;**
* **if (j != i / j) { sum += (i / j) \* (i / j); }**
* **}**
* **}**
* **if (Math.Sqrt(sum) % 1 == 0)**
* **{**
* **output += String.Format("[{0}, {1}], ", i, sum);**
* **}**
* **}**
* **if (output == "[") { return "[]"; }**
* **return output.Substring(0, output.Length - 2) + "]";**
* **}**

**}**

* + - Best Practices4
    - Clever2
  + 0
  + [Fork](https://www.codewars.com/kumite/new?group_id=55ba4c64395226dbac000047&review_id=55aa15eec1eba8a65a000140)
  + Compare with your solution
  + [Link](https://www.codewars.com/kata/reviews/55aa15eec1eba8a65a000140/groups/55ba4c64395226dbac000047)
* [DavidCipriano](https://www.codewars.com/users/DavidCipriano)
* **using System;**
* **using System.Collections.Generic;**
* **public class SumSquaredDivisors**
* **{**
* **public static string listSquared(long m, long n)**
* **{**
* **var result = new List<string>();**
* **for (long number = m; number <= n; number++)**
* **{**
* **var divisors = new List<int>();**
* **long sum = 0;**
* **for (int div = 1; div <= number; div++)**
* **{**
* **if (number % div == 0)**
* **sum += (div \* div);**
* **}**
* **if (Math.Sqrt(sum) % 2 == 0 || sum == 1)**
* **result.Add(string.Format("[{0}, {1}]",number,sum));**
* **}**
* **return string.Format("[{0}]", string.Join(", ", result));**
* **}**
* **}**
  + - Best Practices4
    - Clever0
  + 0
  + [Fork](https://www.codewars.com/kumite/new?group_id=59ca2b518e56b1db20000fa2&review_id=55aa15eec1eba8a65a000140)
  + Compare with your solution
  + [Link](https://www.codewars.com/kata/reviews/55aa15eec1eba8a65a000140/groups/59ca2b518e56b1db20000fa2)
* [myjinxin2015](https://www.codewars.com/users/myjinxin2015)
* **using System;**
* **using System.Linq;**
* **public class SumSquaredDivisors**
* **{**
* **public static int he(int x){**
* **int xx=(int)(Math.Sqrt(x));**
* **return Enumerable.Range(1,xx).Where(y=>x%y==0).Aggregate(0,(a,b)=>a+b\*b+(x/b)\*(x/b))-((Math.Sqrt(x)%1==0)? x : 0);**
* **}**
* **public static string listSquared(long m, long n)**
* **{**
* **return "["+string.Join(", ",Enumerable.Range(Convert.ToInt32(m),Convert.ToInt32(n)).Where(x=>Math.Sqrt(he(x))%1==0&x<n).Select(x=>"["+x.ToString()+", "+he(x)+"]").ToArray())+"]";**
* **}**
* **}**
  + - Best Practices3
    - Clever10
  + 5
  + [Fork](https://www.codewars.com/kumite/new?group_id=564ecb06f0e1d28abd0000f1&review_id=55aa15eec1eba8a65a000140)
  + Compare with your solution
  + [Link](https://www.codewars.com/kata/reviews/55aa15eec1eba8a65a000140/groups/564ecb06f0e1d28abd0000f1)
* [andzej.maciusovic](https://www.codewars.com/users/andzej.maciusovic)
* **using System;**
* **using System.Collections;**
* **using System.Collections.Generic;**
* **public class SumSquaredDivisors**
* **{**
* **public static string listSquared(long from, long to)**
* **{**
* **var parts = new List<string>();**
* **for(var i = from; i <= to; i++) {**
* **var sum = getSquaredDivisorsSum(i);**
* **if(isPerfectSquare(sum)) parts.Add($"[{i}, {sum}]");**
* **}**
* **return "["+string.Join(", ",parts)+"]";**
* **}**
* **static bool isPerfectSquare(long sum) => Math.Sqrt(sum) % 1 == 0;**
* **static long getSquaredDivisorsSum(long n) {**
* **long sum = 1;**
* **long i = 2;**
* **if(n > 1) sum += n\*n;**
* **while(i <= Math.Sqrt(n))**
* **{**
* **if(n % i ==0)**
* **{**
* **sum += i\*i;**
* **long others = (n / i);**
* **if (i != others) sum += others\*others;**
* **}**
* **i++;**
* **}**
* **return sum;**
* **}**

**}**