Let's say that number a feels *comfortable* with number b if a ≠ b and b lies in the segment [a - s(a), a + s(a)], where s(x) is the sum of x's digits.

How many pairs (a, b) are there, such that a < b, both a and b lie on the segment [L, R], and each number feels *comfortable* with the other?

**Example**

For L = 10 and R = 12, the output should be  
comfortableNumbers(L, R) = 2.

Here are all values of s(x) to consider:

* s(10) = 1, so 10 is *comfortable* with 9and 11;
* s(11) = 2, so 11 is *comfortable* with 9, 10, 12 and 13;
* s(12) = 3, so 12 is *comfortable* with 9, 10, 11, 13, 14 and 15.

Thus, there are 2 pairs of numbers *comfortable*with each other within the segment [10; 12]: (10, 11) and (11, 12).

**Input/Output**

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

*Constraints:*  
1 ≤ L ≤ R ≤ 1000.

* **[input] integer R**

*Constraints:*  
1 ≤ L ≤ R ≤ 1000.

* **[output] integer**

The number of pairs satisfying all the above conditions.

<https://codefights.com/arcade/code-arcade/labyrinth-of-nested-loops/pNfGgNk46YZ4c4RW5>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static int S(int n)

{

return n.ToString().Sum(f => f - '0');

}

static List<int> comodos(int a)

{

int izq = a - S(a);

int der = a + S(a);

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

for (int i = izq; i <= der; i++)

{

if (i != a)

{

lista.Add(i);

}

}

return lista;

}

static bool sonConfortables(int a, int b)

{

List<int> comodosA = comodos(a);

List<int> comodosB = comodos(b);

if (comodosB.Contains(a) && comodosA.Contains(b))

{

return true;

}

return false;

}

static int comfortableNumbers(int L, int R)

{

int ans = 0;

for (int i = L; i + 1 <= R; i++)

{

for (int j = i + 1; j <= R; j++)

{

if (sonConfortables(i, j))

{

ans++;

}

}

}

return ans;

}

static void Main(string[] args)

{

//Console.WriteLine(sum(1234));

Console.WriteLine( comfortableNumbers(12,108) );

Console.ReadLine();

}

}

}