**The Prime Cells**

Max. Marks: 100

You are given a grid of size n×nn×n filled with numbers in each of its cells. Now you need to count total cells in the grid such that the sum of the numbers on its top , left , right and bottom cells is a [prime](https://en.wikipedia.org/wiki/Prime_number) number. In case there is no cell in a particular direction assume the number to be as 0.   
**Input**  
First line contains a number nn as input denoting size of the grid. Next nn lines contain nn numbers each denoting value of the elements of the grid in each row.  
**Output**  
In the output you have to give the count of total cells as described above.  
**Constraints**   
2≤n≤1002≤n≤100   
1≤g[i][j]≤1001≤g[i][j]≤100 where g[i][j]g[i][j] is the value in the grid at ithithrow and jthjth column.

**SAMPLE INPUT**

2

1 2

3 4

**SAMPLE OUTPUT**

4

**Explanation**

In the given grid if we pick the first element that is **1** then to its right and bottom are 2 and 3 whose sum is 5 and so its prime. Note that to the left there is no element so we consider it as 0 and same goes for the up direction. Checking this for all yields that all the four cells contribute to the count of cells whose sum of the adjacent cell values is prime.

**Time Limit:**2.0 sec(s) for each input file.

**Memory Limit:**256 MB

**Source Limit:**1024 KB

**Marking Scheme:**Marks are awarded if any testcase passes.

**Allowed Languages:**C, C++, C++14, Clojure, C#, D, Erlang, F#, Go, Groovy, Haskell, Java, Java 8, JavaScript(Rhino), JavaScript(Node.js), Julia, Lisp, Lisp (SBCL), Lua, Objective-C, OCaml, Octave, Pascal, Perl, PHP, Python, Python 3, R(RScript), Racket, Ruby, Rust, Scala, Swift, Visual Basic, Kotlin

<https://www.hackerearth.com/challenge/competitive/july-circuits-17/algorithm/pythagorean-triangles-0158a4c5/>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

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 void Main(string[] args)

{

int n = int.Parse(Console.ReadLine());

int[][] matriz = new int[n][];

for (int i = 0; i < n; i++)

{

matriz[i] = new int[n];

matriz[i] = Array.ConvertAll(Console.ReadLine().Trim().Split(' '), e => int.Parse(e));

}

int ans = 0;

for (int i = 0; i < matriz.Length; i++)

{

for (int j = 0; j < matriz[i].Length; j++)

{

int sum = 0;

if (i - 1 >= 0)

{

sum += matriz[i - 1][j];

}

if (i + 1 < matriz.Length)

{

sum += matriz[i + 1][j];

}

if (j - 1 >= 0)

{

sum += matriz[i][j - 1];

}

if (j + 1 < matriz[i].Length)

{

sum += matriz[i][j + 1];

}

if (esPrimo(sum))

{

ans++;

}

}

}

Console.WriteLine(ans);

Console.ReadLine();

}

}

}