You are given a positive integer N, your task is to calculate the total number of primes less than or equal to N with additional requirement that each digit of those primes should be a prime number itself.

**Example**

For N = 11, the output should be  
primeDigits(11) = 4.  
There are 5 primes less than N or equal to it: 2, 3, 5, 7 and 11. But there is a digit 1 (even two of them) in 11, and 1 is not a prime number. Thus, the answer is 4.

**Input/Output**

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

*Constraints:*  
0 ≤ N ≤ 106

* **[output] integer**

<https://codefights.com/challenge/PYqXBbXgEoDeZmJm3?utm_source=featuredChallenge&utm_medium=email&utm_campaign=email_notification>

//VOLVERLO A HACER CHEQUEANDO PRIMERO LA SEGUNDA CONDICION

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 int primeDigits(int N)

{

int ans = 0;

int[] dig = { '2', '3', '5', '7' };

for (int i = 2; i <= N; i++)

{

if (esPrimo(i))

{

string istr = i.ToString();

int j;

for (j = 0; j < istr.Length; j++)

{

if (!dig.Contains(istr[j]))

{

break;

}

}

if (j == istr.Length)

{

ans++;

}

}

}

return ans;

}

//static int primeDigits(int N)

//{

// int ans = 0;

// int[] dig = { '2', '3', '5', '7' };

// for (int i = 2; i <= N; i++)

// {

// if (esPrimo(i))

// {

// string istr = i.ToString();

// int j;

// for ( j = 0; j < istr.Length; j++)

// {

// if (!dig.Contains(istr[j]))

// {

// break;

// }

// }

// if (j == istr.Length)

// {

// ans++;

// }

// }

// }

// return ans;

//}

static void Main(string[] args)

{

for (int i = 1; i < 1000; i++)

{

int actual = primeDigits(i);

int cont = 0;

while (primeDigits(i) == actual)

{

i++;

cont++;

//Console.Write(primeDigits(i) + " ");

}

Console.Write(cont + " ");

}

// string s = "12345";

Console.ReadLine();

}

}

}

//----------------------MAS EFICIENTE--------------------------------

static int primeDigits(int N)

{

int[] primos =

{

2, 3, 5, 7, 23, 37, 53, 73, 223, 227, 233, 257, 277, 337,

353, 373, 523, 557, 577, 727, 733, 757, 773, 2237, 2273, 2333,

2357, 2377, 2557, 2753, 2777, 3253, 3257, 3323, 3373, 3527,

3533, 3557, 3727, 3733, 5227, 5233, 5237, 5273, 5323, 5333,

5527, 5557, 5573, 5737, 7237, 7253, 7333, 7523, 7537, 7573,

7577, 7723, 7727, 7753, 7757, 22273, 22277, 22573, 22727,

22777, 23227, 23327, 23333, 23357, 23537, 23557, 23753, 23773,

25237, 25253, 25357, 25373, 25523, 25537, 25577, 25733, 27253,

27277, 27337, 27527, 27733, 27737, 27773, 32233, 32237, 32257,

32323, 32327, 32353, 32377, 32533, 32537, 32573, 33223, 33353,

33377, 33533, 33577, 33757, 33773, 35227, 35257, 35323, 35327,

35353, 35527, 35533, 35537, 35573, 35753, 37223, 37253, 37273,

37277, 37337, 37357, 37537, 37573, 52223, 52237, 52253, 52553,

52727, 52733, 52757, 53233, 53323, 53327, 53353, 53377, 53527,

53773, 53777, 55333, 55337, 55373, 55733, 57223, 57373, 57527,

57557, 57727, 57737, 57773, 72223, 72227, 72253, 72277, 72337,

72353, 72533, 72577, 72727, 72733, 73237, 73277, 73327, 73523,

73553, 73727, 73757, 75223, 75227, 75253, 75277, 75323, 75337,

75353, 75377, 75527, 75533, 75553, 75557, 75577, 75773, 77237,

77323, 77377, 77527, 77557, 77573, 77723, 77773, 222323, 222337,

222527, 222533, 222553, 222557, 222773, 223253, 223273, 223277,

223337, 223577, 223753, 223757, 225223, 225227, 225257, 225353,

225373, 225523, 225527, 225733, 227233, 227257, 227377, 227533,

227537, 232333, 232357, 232523, 232753, 232777, 233323, 233327,

233353, 233357, 233557, 233777, 235273, 235337, 235523, 235537,

235553, 235577, 235723, 237233, 237257, 237277, 237373, 237733,

237737, 252223, 252233, 252253, 252277, 252323, 252533, 252727,

252737, 253273, 253537, 253553, 253573, 253733, 253777, 255253,

255523, 255733, 255757, 257273, 257353, 272227, 272257, 272333,

272353, 272533, 272537, 272737, 272777, 273233, 273253, 273323,

273527, 273727, 273773, 275227, 275323, 275357, 275573, 275773,

277223, 277273, 277373, 277577, 277757, 322237, 322327, 322523,

322537, 322573, 322727, 322757, 323233, 323273, 323333, 323377,

323537, 325333, 325537, 325723, 325753, 325777, 327277, 327337,

327553, 327557, 327737, 327757, 332273, 332573, 333227, 333233,

333253, 333323, 333337, 333533, 333737, 333757, 335273, 335323,

335527, 335557, 337223, 337277, 337327, 337537, 352237, 352273,

352327, 352333, 352357, 352523, 352753, 352757, 353237, 353333,

353527, 353557, 353737, 353777, 355573, 355723, 355753, 355777,

357353, 357377, 357727, 357733, 357737, 372223, 372277, 372353,

372377, 372523, 372733, 372773, 373273, 373327, 373357, 373553,

373753, 373757, 373777, 375223, 375227, 375233, 375253, 375257,

375373, 375523, 375527, 375533, 375553, 375757, 375773, 377257,

377327, 377353, 377527, 377537, 377557, 377737, 522227, 522233,

522323, 522337, 522373, 522523, 522553, 522737, 522757, 523333,

523357, 523553, 523573, 523577, 523777, 525253, 525257, 525353,

525373, 525377, 525533, 525727, 525773, 527237, 527273, 527327,

527333, 527353, 527377, 527533, 527557, 527753, 532277, 532327,

532333, 532373, 532523, 532537, 532733, 532757, 533227, 533237,

533257, 533327, 533353, 533573, 533723, 533737, 533777, 535237,

535273, 535333, 535523, 535573, 535727, 535757, 537233, 537373,

537527, 537773, 552353, 552523, 552527, 552553, 552757, 553253,

553277, 553573, 553727, 553733, 553757, 555253, 555257, 555277,

555337, 555523, 555557, 557273, 557377, 557533, 557537, 557573,

572233, 572323, 572333, 572357, 572573, 572777, 573253, 573277,

573523, 573527, 573557, 573737, 573757, 575257, 575557, 575573,

575723, 575753, 575777, 577327, 577333, 577523, 577537, 577573,

577757, 722237, 722257, 722273, 722333, 722353, 722377, 722537,

722723, 722737, 723227, 723257, 723337, 723353, 723553, 723727,

725273, 725323, 725327, 725357, 725537, 725723, 725737, 727273,

727327, 727577, 727733, 727777, 732233, 732257, 732323, 732373,

732533, 733237, 733273, 733277, 733333, 733373, 733753, 733757,

735337, 735373, 735533, 735557, 735733, 737327, 737353, 737533,

737537, 737573, 737753, 737773, 752273, 752527, 753257, 753353,

753373, 753527, 753737, 753773, 755233, 755257, 755273, 755333,

755357, 755737, 757327, 757553, 757577, 757727, 757753, 772273,

772333, 772537, 772573, 772757, 773273, 773533, 773537, 773723,

773777, 775237, 775273, 775553, 775573, 775757, 775777, 777277,

777353, 777373, 777737,

};

int ans = 0;

foreach(int elem in primos) //for (int i = 0; primos[i] <= N; i++)

{

//if (elem >= N) break;

//ans++;

if (elem > N) break;

if (elem == N)

{

ans++;

break;

}

ans++;

}

return ans;

}