You are given an array primes, which consists of all but one prime numbers from some range. Your task is to find the missing number and return it.

It is guaranteed that the answer always exists.

**Example**

For primes = [11, 3, 7], the output should be  
successivePrimes(primes) = 5.

The primes array consists of prime integers in the range [3, 11]. The prime 5 is missing from the array, so 5 is the answer.

**Input/Output**

* **[time limit] 3000ms (cs)**
* **[input] array.integer primes**

A sequence of different prime integers.

*Guaranteed constraints:*  
2 < primes.length ≤ 105,  
2 ≤ primes[i] ≤ 106.

* **[output] integer**

The single missing prime number from primes.

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

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static int successivePrimes(int[] primes)

{

Array.Sort(primes);

int n = primes[primes.Length - 1];

bool[] prime = new bool[n + 1];

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

prime[i] = true;

for (int p = 2; p \* p <= n; p++)

{

// If prime[p] is not changed, then it is a prime

if (prime[p] == true)

{

// Update all multiples of p

for (int i = p \* 2; i <= n; i += p)

prime[i] = false;

}

}

int ans = 0;

// Print all prime numbers

for (int i = primes[0]; i <= n; i++)

{

// if (prime[i] == true)

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

if (prime[i])

{

if ( Array.BinarySearch( primes ,i) < 0)

{

ans = i;

break;

}

}

}

return ans;

}

static void Main(string[] args)

{

//sieveOfEratosthenes(11);

//int[] primes = { 11, 3, 7 };

int[] primes= {2, 5, 7, 11, 13, 17, 19, 23, 3, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541};

Console.WriteLine(successivePrimes(primes));

Console.ReadLine();

}

}

}

----SOLUCION EN PYTHON SIN BINARY SEARCH--------------

def **successivePrimes**(primes):

n = max(primes)

prime = [True for i in range(n+1)]

p=2

while(p \* p <= n):

# If prime[p] is not changed, then it is a prime

if (prime[p] == True):

# Update all multiples of p

for i in range(p \* 2, n+1, p):

prime[i] = False

p+=1

lis =[]

# Print all prime numbers

sumSieve =0

for p in range(min(primes), n+1):

if prime[p]: sumSieve += p

#print p,

return sumSieve - sum(primes)

print successivePrimes([11, 3, 7])