Return the number of ways to write the given n as the sum of two prime numbers.

**Example**  
For n = 20, the output should be  
gold(n) = 2.  
There are two ways to represent 20 as the sum of two prime numbers:

* 3 + 17 = 20,
* 7 + 13 = 20.

**Input/Output**

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

*Guaranteed Constraints:*  
2 ≤ n ≤ 105.

* **[output] integer**

<https://codefights.com/challenge/ypaJW47zSBpJ3b7EJ/solutions>

------------NO ACEPTADO------------------

static List<int> SieveOfEratosthenes(int n)

{

// Create a boolean array "prime[0..n]" and initialize

// all entries it as true. A value in prime[i] will

// finally be false if i is Not a prime, else true.

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

//memset(prime, true, sizeof(prime));

for (int i = 0; i < n + 1; 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;

}

}

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

// Print all prime numbers

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

if (prime[p])

{

//cout << p << " ";

primos.Add(p);

}

return primos;

}

static int gold(int n)

{

List<int> primos = SieveOfEratosthenes(n);

int cont = 0;

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

{

for (int j = i + 1; j < primos.Count; j++)

{

if (primos[i] + primos[j] == n)

{

cont++;

}

}

}

return cont;

}

------------aceptado----------------

int gold(int n)

{

int count = 0;

//Create a list containing elements that are prime numbers smaller than n

var myList = new List<int>();

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

{

if (prime(i)) myList.Add(i);

}

//Convert list to array

int[] arr = myList.ToArray();

for (int i = 0; i < arr.Length - 1; i++)

{

for (int j = i; j < arr.Length; j++)

{

if (arr[i] + arr[j] == n) count++;

}

}

return count;

}

bool prime(int n)

{

if (n < 2) return false;

if (n == 2) return true;

for (int i = 2; i <= Math.Sqrt(n); i++)

{

if (n % i == 0) return false;

}

return true;

}