Author

[Loc\_Nguyen](https://codefights.com/profile/Loc_Nguyen)

https://codefights.com/img/coins_new.png2000

You're given an array arr. Apply the following algorithm to it:

* find intervals of consecutive prime numbers and consecutive non-prime numbers;
* replace each such interval with the sum of numbers in it;
* if the resulting array is different from the initial one, repeat the process;
* otherwise return the resulting array.

Note: here non-prime numbers include 0, 1 and all negative numbers.

**Example:**

* simplifiedArray([1, 2, 3, 5, 6, 4, 2, 3]) = [21, 5]
  + [1,2,3,5,6,4,2,3] --> [1, 2 + 3 + 5, 6 + 4, 2 + 3] --> [1, 10, 10, 5]
  + [1, 10, 10, 5] --> [1 + 10 + 10, 5] --> [21,5]
* simplifiedArray([-3, 4, 5, 2, 0, -10]) = [1, 7, -10]
* **[input] array.integer arr**
  + A non-empty array.  
    1 ≤ |arr| ≤ 1000  
    abs(arr[i]) ≤ 10000
* **[output] array.integer**
  + The resulting array.

<https://codefights.com/challenge/BaYjbaXQ4KZF7GGYJ>

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 List<int> Agrupar(List<int> arr)

{

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

int i = 0;

while (i < arr.Count)

{

int sum = 0;

int j = i;

bool paso = false;

/\* mientras arr[j] sea par lo sumo, cuando deja de serlo agrego la suma al vector \*/

while (j < arr.Count && esPrimo(arr[j])) // arr[j] % 2 == 0)

{

paso = true;

sum += arr[j];

j++;

}

if (paso) agr.Add(sum); /\* si paso por el while lo agrego, ya que sino agrega un cero de mas \*/

if (j == arr.Count) return agr;

sum = 0;

paso = false;

/\* mientras arr[j] sea impar lo sumo, cuando deja de serlo agrego la suma al vector \*/

while (j < arr.Count && ! esPrimo(arr[j])) //arr[j] % 2 != 0)

{

paso = true;

sum += arr[j];

j++;

}

if (paso) agr.Add(sum); /\* si paso por el while lo agrego, ya que sino agrega un cero de mas \*/

i = j; //i toma el valor adelantado de j, (donde termina cada grupo)

}

return agr;

}

static int[] simplifiedArray(int[] arr)

{

List<int> agrup = Agrupar(arr.ToList());

int len = agrup.Count;

do

{

len = agrup.Count;

agrup = Agrupar(agrup);

} while (agrup.Count != len);

return agrup.ToArray();

}