Given a set s of distinct integers and a number k, find the size of the maximum subset of s, such that the sum of any two elements in it is not divisible by the given k.

**Example**

For s = [1, 3, 6, 7] and k = 4, the output should be  
maxNonDivisibleSubset(s, k) = 3.

The largest possible subset of s that satisfies the condition above is [3, 6, 7], because no two integers in it have a sum that is divisible by 4:

* 3 + 6 = 9, and 9 % 4 = 1;
* 3 + 7 = 10, and 10 % 4 = 2;
* 6 + 7 = 13, and 13 % 4 = 1.

**Input/Output**

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

An array of unique elements.

*Guaranteed constraints:*  
1 ≤ s.length ≤ 105,  
1 ≤ si ≤ 109.

* **[input] integer k**

*Guaranteed constraints:*  
1 ≤ k ≤ 100.

* **[output] integer**

The size of the maximum subset of s, such that the sum of any two elements in it is not divisible by k.

<https://codefights.com/challenge/dZTFsam7DZgzbCyey?utm_source=emailNotification&utm_medium=email&utm_campaign=featuredChallenge>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication2

{

class Program

{

static int maxNonDivisibleSubset(int[] s, int k)

{

//int N, K, n;

int N = s.Length;

//cin >> N >> K;

//int occ[K] = {0};

int[] occ = new int[k];

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

{

occ[i] = 0;

}

int res = 0;

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

{

//cin >> n;

occ[s[i] % k]++;

}//for

for (int i = 1; i < (k + 1) / 2; i++)

res += (occ[i] > occ[k - i] ? occ[i] : occ[k - i]);

//res += (occ[0] >= 1) + (k % 2 == 0 && occ[k / 2]>0) );

res += (occ[0] >= 1) ? 1 : 0;

res += (k % 2 == 0 && occ[k / 2] > 0) ? 1 : 0;

return res;

}

static void Main(string[] args)

{

int[] s = {1, 3, 6, 7};

int k = 4;

Console.WriteLine(maxNonDivisibleSubset(s,k));

Console.ReadLine();

}

}

}