1. An array is defined to be a **235 array** if the number of elements divisible by 2 plus the number of elements divisible by 3 plus the number of elements divisible by 5 plus the number of elements not divisible by 2, 3, or 5 is equal to the number of elements of the array. Write a method named **is123Array** that returns 1 if its array argument is a 235 array, otherwise it returns 0.

If you are writing in Java or C#, the function signature is

int is235Array(int[ ] a)

If you are writing in C or C++, the function signature is

int is235Array(int a[ ], int len) where len is the length of a

Hint: remember that a number can be divisible by more than one number

Examples

In the following: <**a, b, c, d**> means that the array has **a** elements that are divisible by 2, **b** elements that are divisible by 3, **c** elements that are divisible by 5 and **d** elements that are not divisible by 2, 3, or 5.

if a is	**return**	**reason**
{2, 3, 5, 7, 11}	1	because one element is divisible by 2 (a[0]), one is divisible by 3 (a[1]), one is divisible by 5 (a[2]) and two are not divisible by 2, 3, or 5 (a[3] and a[4]). So we have <1, 1, 1, 2> and 1+1+1+2 == the number of elements in the array.
{2, 3, 6, 7, 11}	0	because two elements are divisible by 2 (a[0] and a[2]), two are divisible by 3 (a[1] and a[2]), none are divisible by 5 and two are not divisible by 2, 3, or 5 (a[3] and a[4]). So we have <2, 2, 0, 2> and 2 + 2 + 0 + 2 == 6 != the number of elements in the array.
{2, 3, 4, 5, 6, 7, 8, 9, 10}	0	because <5, 3, 2, 1> and 5 + 3 + 2 + 1 == 11 != the number of elements in the array.
{2, 4, 8, 16, 32}	1	because <5, 0, 0, 0> and 5 + 0 + 0 + 0 == 5 == the number of elements in the array.
{3, 9, 27, 7, 1, 1, 1, 1, 1}	1	because <0, 3, 0, 6> and 0 + 3 + 0 + 6 == 9 == the number of elements in the array.
{7, 11, 77, 49}	1	because <0, 0, 0, 4> and 0 + 0 + 0 + 4 == 4 == the number of elements in the array.

{2}	1	because <1, 0, 0, 0> and 1 + 0 + 0 + 0 == 1 == the number of elements in the array.
{}	1	because <0, 0, 0, 0> and 0 + 0 + 0 + 0 == 0 == the number of elements in the array.
{7, 2, 7, 2, 7, 2, 7, 2, 3, 7, 7}	1	because <4, 1, 0, 6> and 4 + 1 + 0 + 6 == 11 == the number of elements in the array.

2. The Fibonacci sequence of numbers is 1, 1, 2, 3, 5, 8, 13, 21, 34, ... The first and second numbers are 1 and after that $n_i = n_{i-2} + n_{i-1}$, e.g., 34 = 13 + 21. A number in the sequence is called a Fibonacci number. Write a method with signature **int closestFibonacci(int n)** which returns the largest Fibonacci number that is less than or equal to its argument. For example, closestFibonacci(12) returns 8 because 8 is the largest Fibonacci number less than 12 and closestFibonacci(33) returns 21 because 21 is the largest Fibonacci number that is <= 33. closestFibonacci(34) should return 34. If the argument is less than 1 return 0. Your solution must **not** use recursion because unless you cache the Fibonacci numbers as you find them, the recursive solution recomputes the same Fibonacci number many times.

Copy and paste your answer here and click the "Submit answer" button

3. An array a is defined to be **self-referential** if for i=0 to a.length-1, a[i] is the count of the number of times that the value i appears in the array. As the following table indicates, {1, 2, 1, 0} is a self-referential array.

i	a[i]	comment
0	1	There is one 0 in the array. (a[0] = 1)
1	2	There are two 1s in the array (a[1] = 2)
2	1	There is one 2 in the array (a[2] = 1)
3	0	There are no 3s in the array (a[3] = 0)

Here are some examples of arrays that are not self-referential:

{2, 0, 0} is not a self-referential array. There are two 0s and no 1s. But unfortunately there is a 2 which contradicts a[2] = 0.

{0} is not a self-referential array because there is one 0, but since a[0] = 0, there has to be no 0s.

{1} is not a self-referential array because there is not a 0 in the array as required by a[0] = 1.

Self-referential arrays are rare. Here are the self-referential arrays for arrays of lengths up to 10 elements:

{1, 2, 1, 0} (see above)
{2, 0, 2, 0} (there are two 0s, no 1s, two 2s and no 3s
{2, 1, 2, 0, 0} (there are two 0s, one 1, two 2s, no 3s and no 4s)
{3, 2, 1, 1, 0, 0, 0} (there are three 0s, two 1s, one 2, one 3, no 4s, 5s or 6s)
{4, 2, 1, 0, 1, 0, 0, 0} (there are four 0s, two 1s, one 2, no 3s, one 4, and no 5s, 6s, or 7s)
{5, 2, 1, 0, 0, 1, 0, 0, 0} (there are five 0s, two 1s, one 2, no 3s or 4s, one 5, and no 6s, 7s, or 8s)
{6, 2, 1, 0, 0, 0, 1, 0, 0, 0} (there are six 0s, two 1s, one 2, no 3s, 4s, or 5s, one 6, and no 7s, 8s, or 9s)

Write a function named *isSelfReferential* that returns 1 if its array argument is self-referential, otherwise it returns 0.

If you are programming in Java or C#, the function signature is

```
int isSelfReferential(int[ ] a)
```

If you are programming in C or C++, the function signature is

```
int isSelfReferential(int a[ ], int len)
```
where len is the number of elements in the array

Copy and paste your answer here and click the "Submit answer" button