Find the Access Codes  
=====================  
  
In order to destroy Commander Lambda's LAMBCHOP doomsday device, you'll need access to it. But the only door leading to the LAMBCHOP chamber is secured with a unique lock system whose number of passcodes changes daily. Commander Lambda gets a report every day that includes the locks' access codes, but only the Commander knows how to figure out which of several lists contains the access codes. You need to find a way to determine which list contains the access codes once you're ready to go in.   
  
Fortunately, now that you're Commander Lambda's personal assistant, Lambda has confided to you that all the access codes are "lucky triples" in order to make it easier to find them in the lists. A "lucky triple" is a tuple (x, y, z) where **x divides y** and y divides z, such as (1, 2, 4). With that information, you can figure out which list contains the number of access codes that matches the number of locks on the door when you're ready to go in (for example, if there's 5 passcodes, you'd need to find a list with 5 "lucky triple" access codes).  
  
Write a function solution(l) that takes a list of positive integers l and counts the number of "lucky triples" of (li, lj, lk) where the list indices meet the requirement i < j < k. The length of l is between 2 and 2000 inclusive. The elements of l are between 1 and 999999 inclusive. The solution fits within a signed 32-bit integer. Some of the lists are purposely generated without any access codes to throw off spies, so if no triples are found, return 0.   
  
For example, [1, 2, 3, 4, 5, 6] has the triples: [1, 2, 4], [1, 2, 6], [1, 3, 6], making the solution 3 total.  
  
Languages  
=========  
  
To provide a Java solution, edit Solution.java  
To provide a Python solution, edit solution.py  
  
Test cases  
==========  
Your code should pass the following test cases.  
Note that it may also be run against hidden test cases not shown here.  
  
-- Java cases --  
Input:  
Solution.solution([1, 1, 1])  
Output:  
    1  
  
Input:  
Solution.solution([1, 2, 3, 4, 5, 6])  
Output:  
    3  
  
-- Python cases --  
Input:  
solution.solution([1, 2, 3, 4, 5, 6])  
Output:  
    3  
  
Input:  
solution.solution([1, 1, 1])  
Output:  
    1