'''

In a Mathematics class students are given a list which are distinct and

positive integers. Students should do the calculation and return

the number of quadraples (p, q, r, s) such that p \* q = r \* s

where p, q, r, and s are elements of list, and p != q != r != s.

Input Format

-------------

Line-1: An integer N, number of elements in the list

Line-2: N space separated integers, the list[]

Output Format

-------------

An integer which is no of quadraples

Sample Input-1:

---------------

4

2 3 4 6

Sample Output-1:

----------------

8

Explanation:

------------

There are 8 valid quadraples:

(2,6,3,4) , (2,6,4,3) , (6,2,3,4) , (6,2,4,3)

(3,4,2,6) , (4,3,2,6) , (3,4,6,2) , (4,3,6,2)

Sample Input:

-------------

5

1 2 4 8 16

Sample Output:

--------------

24

Explanation:

------------

There are 24 valid quadraples:

(1,16,2,8) , (1,16,8,2) , (16,1,2,8) , (16,1,8,2)

(2,8,1,16) , (2,8,16,1) , (8,2,1,16) , (8,2,16,1)

(1,8,4,2) , (1,8,2,4) , (8,1,4,2) , (8,1,2,4)

(4,2,1,8) , (4,2,8,1) , (2,4,1,8) , (2,4,8,1)

(2,16,4,8) , (2,16,8,4) , (16,2,4,8) , (16,2,8,4)

(4,8,2,16) , (4,8,16,2) , (8,4,2,16) , (8,4,16,2)

'''

n=int(input())

l=list(map(int,input().split()))

d=dict()

for i in range(0,len(l)):

for j in range(i+1,len(l)):

val=l[i]\*l[j]

if val in d:

d[val]+=1

else:

d[val]=1

count=0

for i in d:

if(d[i]>1):

count+=4\*(d[i])\*(d[i]-1)

print(count)

In a Marketing Agency, each agent will mentor either two sub-agents,

or zero agents. Now, based on ranks given to sub-agents, the mentor agent

will be ranked with the top rank among two sub-agents.

The ranks are in the range [1,20], More than one agent can have same rank.

At the end, all mentor agents and sub agents, will be treated as agents only.

You are given the entire ranking picture as a tree.

Your task is to find out second top agen2 5 2 -1 -1 2 4

t in the Marketing agency and

print the agent's rank, If no such agent exist, print -2.

Implement the class Solution:

1. public int findSecondTopAgent(BinaryTreeNode root): returns an integer.

NOTE:

- In the input '-1', indicates empty(null).

Input Format:

-------------

A single line of space separated integers, ranks of each individual.

Output Format:

--------------

Print an integer, second top agent based on rank.

Sample Input-1:

---------------

2 5 2 -1 -1 2 4

Sample Output-1:

----------------

4

Sample Input-2:

---------------

3 3 3 3 3

Sample Output-2:

----------------

-2

::For Tree structure refer to Hint::

import java.util.\*;

/\*

class Node {

public int data;

public Node left;

public Node right;

public Node(int value) {

data = value;

left = null;

right = null;

}

}

\*/

class Solution {

public int findSecondTopAgent(Node root) {

//implement your code here.

ArrayList<Integer> arr=new ArrayList<>();

Node curr=root;

order(curr,arr);

Set<Integer>hs=new HashSet<>(arr);

if(hs.size()<2){

return -2;

}

ArrayList<Integer> arr1=new ArrayList<>(hs);

Collections.sort(arr1);

return arr1.get(1);

}

public void order(Node r,ArrayList<Integer> arr){

if(r==null){

return;

}

// order(r.left,arr);

arr.add(r.data);

// order(r.right,arr);

// }

// if(r.left!=null){

order(r.left,arr);

// }

// if(r.right!=null){

order(r.right,arr);

// }

}

}

Clavius, the person who suggested grouping of data using brackets.

He has suggested that the group of brackets should be Well-Formed.

A string is said to be Well-Formed, if:

- The string is empty.

- The string can be written as MN, (M is appended to N),

where M and N are Well-Formed Strings, or

- The string can be written as [M], where M is Well-Formed string.

You will be given a string B, consists of square brackets only '[' , ']'.

In one operation, you can insert one square bracket at any position of B.

i.e., Given B = [][]], in one operation you can add a open square bracket,

now B can be Well-Formed string, if you add at index 0,1,2,3,4 =>

[[][]], [][[]], [][][], and not a Well-Formed string if you add last postion.

Your task is to return the least number of operations required,

to make B as a Well-Formed string.

Input Format:

-------------

A string, B contains only characters []

Output Format:

--------------

Print an integer value.

Sample Input-1:

---------------

[]]][

Sample Output-1:

----------------

3

Sample Input-2:

---------------

[]][[]]

Sample Output-2:

----------------

1

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

String s=sc.next();

int count=0;

Stack<Character> st=new Stack<>();

for(int i=0;i<s.length();i++){

if(s.charAt(i)==']' && !st.isEmpty()){

st.pop();

}

else if(s.charAt(i)==']' && st.isEmpty()){

count+=1;

}

else{

st.push(s.charAt(i));

}

}

while(!st.isEmpty()){

count+=1;

st.pop();

}

System.out.println(count);

}

}

There are n boxes of balls in a row, every i-th box contains balls[i] balls.

A series of boxes is called Arithmetic, if it consists of at least three boxes

and if the difference between the number of balls in any two consecutive boxes

is same.

A subset of boxes is called Arithmetic Subset,

if subset[i], subset[i+1], subset[i+2], ..., subset[j] is Arithmetic.

Your task is to findout the number of Arithmetic Subsets in the row of boxes.

Input Format:

-------------

Line-1 : An integer N, number of boxes.

Line-2 : N space separated integers balls[i], i-th box contains balls[i] balls.

Output Format:

--------------

Print an integer, number of Arithmetic Subsets.

Sample Input:

---------------

4

1 2 3 4

Sample Output:

----------------

3

Explanation:

------------

Arithmetic Subsets are: [1,2,3], [2,3,4], [1,2,3,4].

Sample Input:

---------------

5

1 3 5 7 9

Sample Output:

----------------

6

Explanation:

------------

Arithmetic Subsets are: [1,3,5], [3,5,7], [5,7,9], [1,3,5,7], [3,5,7,9], [1,3,5,7,9].

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int[] arr=new int[n];

for(int i=0;i<n;i++){

arr[i]=sc.nextInt();

}

int count=0;

int k=-1;

int c=0;

for(int i=0;i<arr.length-2;i++){

if(arr[i+1]-arr[i]==arr[i+2]-arr[i+1]){

c+=1;

count+=c;

}

else{

c=0;

}

}

System.out.println(count);

}

}