PSC - Bruteforce

Count Possible Pairs

Description

You are given a system of equations:

$x^2 + y = n$ $\quad 1000 \geq x \geq 0$

$y^2 + x = m$ $\quad 1000 \geq y \geq 0$

x^2 + y = n

y^2 + x = m

You have to count, how many there are pairs of integers (x, y) (0 ≤ x, y) which satisfy the system.

Input

Input Format :

A single line contains two integers n,m

Constraints :

1≤n,m≤1000

Output

On a single line print the count

$\cdots x^2 + y = 9 \quad \cdots 3^2 + 0 = 9$ ✓

$\cdots y^2 + x = 3 \quad \cdots 0^2 + 3 = 3$ ✓

$x = 3, y = 0$ $\quad (3, 0)$

$x = 2, y = 1$

$x = 0, y = 3$

Sample Input 1

```
9 3
```

Sample Output 1

```
1
```

Hint

Output Explanation :

In the first sample the suitable pair is integers (3,0)

$x^2 + y = n$

$x^2 + y = 1000$

+ve +ve

$y = 1001$

$x^2 + 1001 = 1000$

$y^2 + x = m$

$y^2 + x = 1000$

1. Generate all possible pairs (x, y).
2. For each pair,
   check if it satisfies both equations.
   If yes, count++.

x	y
0	0
0	1
0	2
⋮	⋮
0	1000

x = 0

x	y
1	0
1	1
1	2
⋮	⋮
1	1000

x = 1

x = 2

x	y
1000	0
1000	1
1000	2
⋮	⋮
1000	1000.

x = 1000

```
int count = 0;
for(int x = 0; x ≤ 1000; x++) {
    for(int y = 0; y ≤ 1000; y++) {
        // check if (x, y) satisfies both equations.
        // if yes, count++;
    }
}
```

Math.sqrt(x) + y

√x + y ✗

x * x

$x^2 + y = n$
$y^2 + x = m$

```
int count = 0;
for(int x = 0; x ≤ 1000; x++) {
    for(int y = 0; y ≤ 1000; y++) {
        // check if (x, y) satisfies both equations.
        // if yes, count++;
        if((x*x + y == N) && (y*y + x = M)){
            count++;
        }
    }
}
S.O.P (count);
```

m

n

Sum of Special Pairs

Description

You are given an array `A` of `N` integers.

Write a program to find the `sum` of the absolute difference between all such pairs (A[i], A[j]) such that i < j and (j-i) is prime.

Input

Input Format :

First line contains N, size of array A.

Second line contains N space separated integers which are elements of A

Constraints :

1 <= N <= 1000

Output

Output one number, total number of pairs pairs (A[i], A[j]) such that i < j and (j-i) is prime.

Sample Input 1

```
6
1 2 3 5 7 12
```

Sample Output 1

```
45
```

A | 1 | 2 | 3 | 5 | 7 | 12 |
0 1 2 3 4 5

i	j	j - i		abs(A[i] - A[j])
0	1	1-0=1	✗	
0	2	2-0	✓	abs(1 − 3) = 2 (−2)
0	3	3-0	✓	abs(1 − 5) = 4
0	4	4-0	✗	
0	5	5-0	✓	abs(1 − 12) = 11 (−11)
1	2	2-1=1		
1	3	3-1=2	✓	abs(2 − 5) = 3
1	4	4-1=3	✓	−(2 − 7) = 5
1	5	5-1=4	✗	
2	3			
2	4			
2	5			

2 + 4 + 11 + 3 + ...

Algo:

int sum = 0

to generate all possible pairs (i, j) s.t i < j

- if ((j - i) is prime) {

// sum of abs diff. between A[i] & A[j]

sum += ____

}

A 

```
int sum = 0;                                          n
for (int i = 0; i < A.length; i++) {                  i = n-1
    for (int j = i+1; j < A.length; j++) {            j = n
                                      n
        if (isPrime(j - i)) {
            sum += Math.abs(A[i] - A[j]);
        }
    }
S.O.P(sum);
```

```
public static boolean isPrime(int n) {
    // Logic to check if n is prime.
    // If yes, return true ELSE return false.
}
```

Apply Basic Maths

Edit

Description

You are given an array `A` with `N` elements. You are allowed to `remove only one element` , which makes the `sum of all the remaining elements exactly divisible by 7` .

Your task is to find the `first index of smallest element` that can be removed from array. If there is no an swer print `-1` .

Input

Input Format

The first line contains a single integer `N` .

Next line contains N space separated integers A[k] , (0 <= k < N).

Constraints

1 < N < 100000

0 < A[k] < 1000000000

Output

Output Format

Print a single line containing one integer, the first array index of the smallest element that CAN be removed, and -1 if there is no such element that he can remove!

Sample Input 1

```
5
14 7 8 2 4
```

Sample Output 1

```
1
```

int Sum = 0

min = ~~∞~~ Integer.MAX_VALUE 14

minIdx = ~~-1~~ 0

Compare A[i] & min

A | 14 | 7 | 8 | 2 | 4 |
0 1 2 3 4

totalSum = 14+7+8+2+4
= 35.

		totalSum − A[i]	remSum % 7	
i=0	Remove A[0]=14,	remSum = 35−14 = 21	21 % 7 = 0	✓
i=1	Remove A[1]=7,	remSum = 35−7 = 28	28 % 7 = 0	✓
i=2	Remove A[2]=8	remSum = 35−8 = 27	27 % 7	✗
	Remove A[3]=2	remSum = 35−2 = 33	33 % 7	✗
	Remove A[4]=4	remSum = 35−4 = 31	31 % 7	✗

min : ~~14~~ 7 minIdx : ~~0~~ 1

Algo:

1. Find totalSum (sum of all elements in the array)
2. for i = 0 to n−1 {
   remSum = totalSum − A[i]
   if (remSum % 7 == 0) {
   // Compare A[i] with min
   //
   }
   }

```
1.  int totalSum = 0
    for (int i = 0; i < n; i++)
        totalSum += A[i]

2   int min = Integer.MAX_VALUE;
    int minIdx = -1;
    for (int i = 0; i < n ; i++) {
        int remSum = totalSum - A[i];
        if (remSum % 7 == 0) {
            // compare A[i] with min. If A[i] is smaller, update min
            if (A[i] < min) {
                min = A[i]
                minIdx = i
            }
        }
    }
3.  S.O.P (minIdx);
```

A

14	7	8	2	4
0	1	2	3	4

totalSum = 35

= 35 - 14

totalSum = 21

Masai Palindromic Substring

Edit

Description

You are provided a string `S`.

Write a program that returns length of the longest palindromic substring of that string.

Input

Input Format

First line contains S, a string.

Constraints

`1 <= Length of string <= 100

Output

Output Format

Output one number which is length of the longest substring which is a palindrome

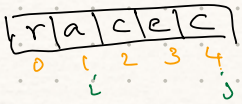
Sample Input 1	Sample Output 1
thisracecarisgood	7

max: ~~0~~ ~~1~~ 2

S = thisracecarisgood

1. Generate all possible substrings of S
2. For each substring:
   1. Check if it is a palindrome
   2. If YES, find its length & compare with max value found so far. if length > max, update max.

```
int max = 0;
for (int i=0; i<S.length(); i++) {
    for (int j=i+1; j<=S.length(); j++) {
        String substr = S.substring(i, j);
        if (isPalindrome(substr)) {
            int len = j-i;
            if (len > max)
                max = len;
        }
    }
}
```



```
s.o.p(max);
```