Given an integer **n**, the task is to define an array **arr[]** of size **n** &

Print the **count of element whose value is equal to its index value**,

For Ex:- if the value, **"4"** is present at **arr[4]** , therefore it would qualify as an **element whose value is equal to its index value**.

Input Format

An integer **n**, which is the size of the array **arr[]**

n integers each in a new line, depicting the elements of the array **arr[]**

Constraints

```
- 0 <= arr.length <= 1000

- 0 <= arr[i] <= 1000
```

Output Format

Single line of output

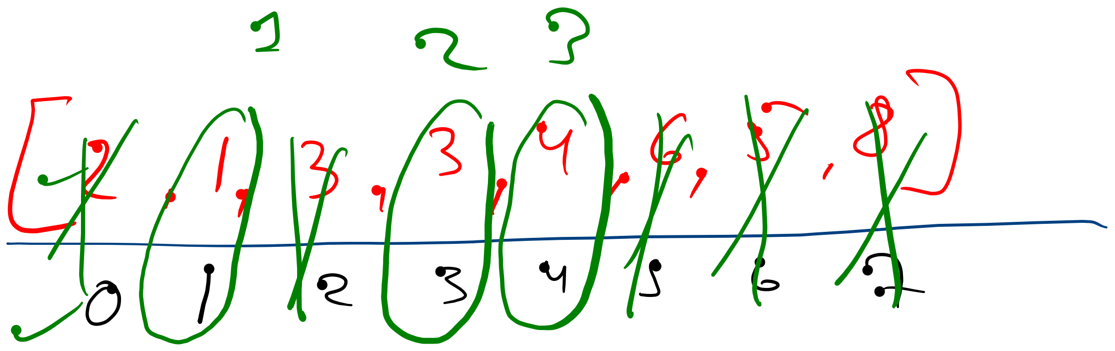
An integer, which is the number of elements in the array, whose value is equal to its index value.

Sample Input 0

```
5
4
1
5
3
5
```

Sample Output 0

```
2
```



ans = 3

Submitted Code

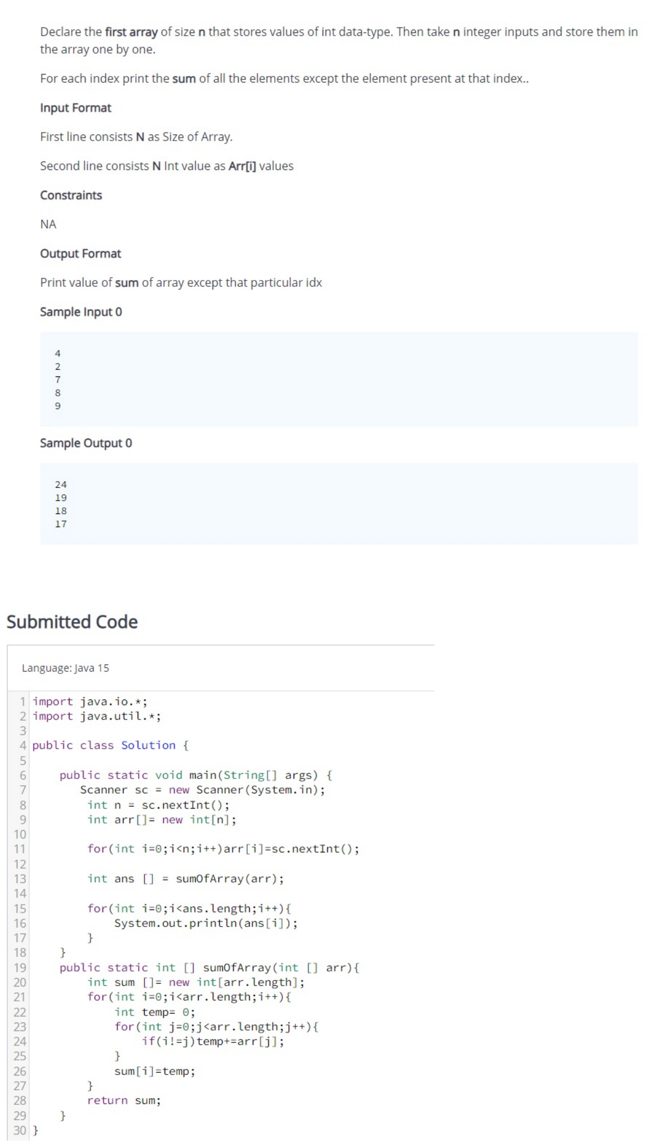
Language: Java 15

```java
import java.io.*;
import java.util.*;

public class Solution {

    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();

        System.out.print(elementIndex(arr));
    }
    public static int elementIndex(int arr[]){
        int count=0;
        for(int i=0;i<arr.length;i++){
            if(arr[i]==i)count++;
        }
        return count;
    }
}
```



Declare the **first array** of size **n** that stores values of int data-type. Then take **n** integer inputs and store them in the array one by one.

For each index print the **sum** of all the elements except the element present at that index..

Input Format

First line consists **N** as Size of Array.

Second line consists **N** Int value as **Arr[i]** values

Constraints

NA

Output Format

Print value of **sum** of array except that particular idx

Sample Input 0

```
4
2
7
8
9
```

Sample Output 0

```
24
19
18
17
```

Submitted Code

Language: Java 15

```java
import java.io.*;
import java.util.*;

public class Solution {

    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 ans [] = sumOfArray(arr);

        for(int i=0;i<ans.length;i++){
            System.out.println(ans[i]);
        }
    }
    public static int [] sumOfArray(int [] arr){
        int sum []= new int[arr.length];
        for(int i=0;i<arr.length;i++){
            int temp= 0;
            for(int j=0;j<arr.length;j++){
                if(i!=j)temp+=arr[j];
            }
            sum[i]=temp;
        }
        return sum;
    }
}
```

[2, 7, 8, 9]

[24, 19, 18, 17]

[1, , ,]

```
for(i=1; i<n-1; i++) —
    output[i] = output[i-1] + arr[i-1]
product =
for(i=n-1; i>=0; i--) —
    output[i] = output[i] * product
    product = arr[i] * product
```

[2, 4, 6, 8]

]

```
for(i=0; i<n; i++){
    int sum = 0;
    for(j=0; j<n; j++){
        if (i != j)
            sum += arr[i];
    }
    arr[i] = sum;
```

Declare the **first array** of size **n** that stores values of int data-type. Then take **n** integer inputs and store them in the array one by one. Print the **minimum** amongst all the elements of the array.

Input Format

First line consists **N** as Size of Array

Second line consists **N** Integer value as **Arr[i]** values

Constraints

NA

Output Format

Print the **Minimum** element in array

Sample Input 0

```
5
10
4
9
55
21
```

Sample Output 0

```
4
```

Explanation 0

4 is the minimum among all these

[10, 4, 9, 55, 21]

4

Integer.MIN_VALUE → Max

Integer.MAX_VALUE → Min

~~min = 0~~

Submitted Code

Language: Java 15

```java
import java.io.*;
import java.util.*;

public class Solution {

    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();

       System.out.print(minVal(arr));
    }

    public static int minVal(int arr[]){
       int min = Integer.MAX_VALUE;

       for(int i=0;i<arr.length;i++){
           if(min>arr[i]){
               min=arr[i];
           }
       }
       return min;
    }
}
```

[-1, 2, 4, 6, 8]

for(- - - -)

if (min < arr[i])

-1

min = arr[i]

-1

Given an integer array **nums** and an integer **val**, remove all **occurrences** of val in nums in-place. The relative order of the elements may be changed.

Since it is impossible to change the length of the array in some languages, you must instead have the result be placed in the first part of the array nums. More formally, if there are k elements after removing the duplicates, then the first k elements of nums should hold the final result. It does not matter what you leave beyond the first k elements.

Return **k** after placing the final result in the first k slots of nums.

Do not allocate extra space for another array. You must do this by modifying the input array in-place with O(1) extra memory.

Input Format

First line of input contains integer **N** as size of array.

Second line of input contains **N** integers representing elements of array.

Third line of input contains integer **val**.

Constraints

```
0 <= N <= 100
0 <= nums[i] <= 50
0 <= val <= 100
```

Output Format

Return the value of **k**.

Sample Input 0

```
4
2 3 2 3
3
```

Sample Output 0

```
2
```

Explanation 0

Your function should return k = 2, with the first two elements of nums being 2. It does not matter what you leave beyond the returned k (hence they are underscores).

(2 3 2 3

K = 3

2

[2, 3, 4, 5, 3, 4, 5, 5, 5, 5, 4, 9
1 2 3 4 5 6 7

K = 5

7

[2, 3, 4, 4, 9, 5

K = 4

2

Submitted Code

Language: Java 15

```java
import java.io.*;
import java.util.*;

public class Solution {

    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 k=sc.nextInt();
        System.out.print(removeK(arr,k));
    }
    public static int removeK(int arr[], int k){
        int count=0;
        for(int i=0;i<arr.length;i++){
            if(arr[i]!=k){
                count++;
            }
        }
        return count;
    }
}
```