

JAVA SAMPLE PROGRAMS

Rajeev works in the data center lab of the survey department. He has been assigned the task of identifying "repeated numbers" in a given set of numbers. He approaches you to help him achieve this.

Given an array of numbers, your task is to return the first repeated number in the array starting from the first index.

For example:

If input1 = 6 representing the number of elements in array, and input2 = {1, 2, 4, 1, 2, 8} representing the given array, then the result should be 1 which is the first repeated number in the array.

Special conditions to be taken care:

Note 1: You should ignore the negative numbers and zeros. The program should consider only non-zero, non-negative numbers from the given array.

Note 2: If no number is repeated then the output should be first element of the array.

Note 3: If all elements in the array are negative or 0's the output should be 0.

For example:

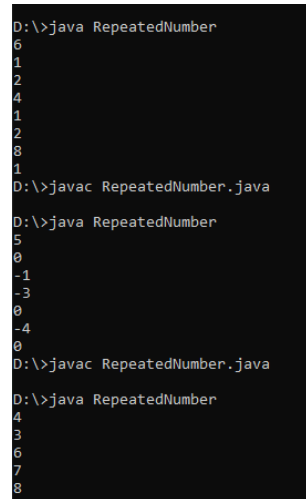
Input	Result
6 1 2 4 1 2 8	1

CODE:

```
import java.util.*;
class RepeatedNumber{
public static void main(String args[])
{
Scanner obj= new Scanner(System.in);
int n=obj.nextInt(); //ARRAY SIZE
int[] a= new int[n];
boolean f=false;
for (int i=0; i<n; i++)
{
a[i]=obj.nextInt();
}
for (int p=0;p<n;p++)
{
if(a[p]>0)
{
f=true;
}else{
System.out.print("0");
break;
}
}
if(f){
for (int j=0; j<n; j++){
for(int k=0; k<n; k++){
```

```
if(a[j]==a[k])
{
System.out.print(a[j]);
break;
}else{
System.out.print(a[0]);
break;
}
}
break;
}
}
}
```

OUTPUT:



```
D:\>java RepeatedNumber
6
1
2
4
1
2
8
1
D:\>javac RepeatedNumber.java
D:\>java RepeatedNumber
5
0
-1
-3
0
-4
0
D:\>javac RepeatedNumber.java
D:\>java RepeatedNumber
4
3
6
7
8
```

Madhav has been assigned the task of finding the sum of all prime numbers in a given array, except the largest prime number in the array. Madhav approaches you to help him do this by writing a program.

Given an array of numbers, you are expected to find the sum of all prime numbers in the given array. You must however exclude the largest prime number while performing this addition.

For example:

If input1 = 11 representing the number of elements in the array, and input2 = {10, 41, 18, 50, 43, 31, 29, 25, 59, 96, 67} representing the given array, then the expected output is 203, which is the sum of all prime numbers in this array except the largest prime number 67.

Explanation:

The prime numbers in this array are 41, 43, 31, 29, 59 and 67.

The largest prime number in this array is 67.

So, let us leave out 67 and add all the other prime numbers to get the output.

Therefore, output = 41 + 43 + 31 + 29 + 59 = 203.

Special conditions to be taken care:

Note: If the array does NOT contain any prime number, the output should be the sum of all numbers in the array except the largest number.

For example, if input1 = 4 representing the number of elements in the array and input2 = {10, 20, 30, 40}, then the expected output = 10 + 20 + 30 = 60.

For example:

Input	Result
11 10 41 18 50 43 31 29 25 59 96 67	203
4 10 20 30 40	60

CODE:

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

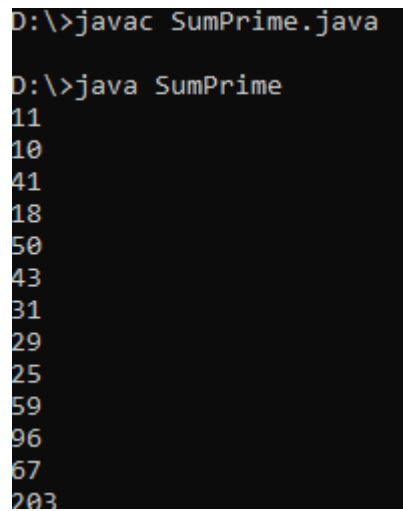
```
class SumPrime {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int n = sc.nextInt();  
        int[] e = new int[n];  
        int sumOfPrimes = 0;  
        int largestPrime = 0; // Track the largest prime encountered
```

```
        for (int i = 0; i < n; i++) {  
            e[i] = sc.nextInt();  
            if (isPrime(e[i])) {  
                sumOfPrimes += e[i];  
                if (e[i] > largestPrime) { // Update largestPrime if needed  
                    largestPrime = e[i];  
                }  
            }  
        }  
    }  
}
```

```
    System.out.println(sumOfPrimes - largestPrime); // Subtract the largest prime  
}
```

```
static boolean isPrime(int num) {  
    if (num <= 1) {  
        return false;  
    }  
    for (int i = 2; i * i <= num; i++) {  
        if (num % i == 0) {  
            return false;  
        }  
    }  
    return true;  
}  
}
```

OUTPUT:



```
D:\>javac SumPrime.java  
D:\>java SumPrime  
11  
10  
41  
18  
50  
43  
31  
29  
25  
59  
96  
67  
203
```

What is a prime number?

A prime number (or a prime) is a natural number greater than 1 that has no positive divisors other than 1 and itself. In other words, a prime number is a whole number greater than 1, whose only two whole number factors are 1 and itself. The first prime numbers are 2, 3, 5, 7, 11, 13, 17, 19, 23 and 29.

Given an array with 'N' elements, you are expected to find the sum of the values that are present in non-prime indexes of the array. Note that the array index starts with 0 i.e. the position (index) of the first array element is 0, the position of the next array element is 1, and so on.

Example 1:

If the array elements are {10, 20, 30, 40, 50, 60, 70, 80, 90, 100}, then the values at the non-prime index are 10, 20, 50, 70, 90, 100 and their sum is 340.

Example 2:

If the array elements are {-1, -2, -3, 3, 4, -7}, then the values at the non-prime index are -1, -2, 4 and their sum is 1.

Example 3:

If the array elements are {-4, -2}, the values at the non-prime index are -4, -2 and their sum is -6.

For example:

Input	Result
10 10 20 30 40 50 60 70 80 90 100	340
6 -1 -2 -3 3 4 -7	1
2 -4 -2	-6

CODE:

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

```
class SumIndexPrime {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] e = new int[n];
        int sum = 0;
```

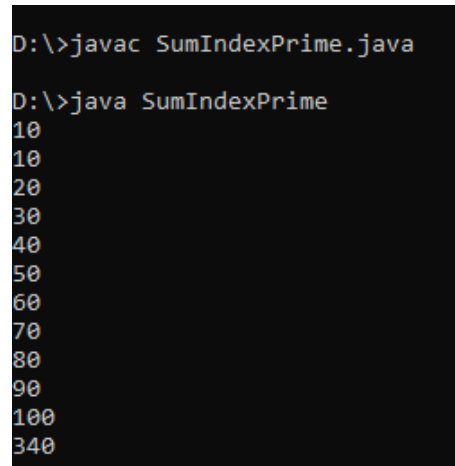
```
        for (int i = 0; i < n; i++) {
            e[i] = sc.nextInt();
            if (!isPrime(i)) {
                sum += e[i];
            }
        }
```

```
        System.out.println(sum);
    }
```

```
    static boolean isPrime(int num) {
        if (num <= 1) {
            return false;
        }
    }
```

```
    for (int i = 2; i * i <= num; i++) {  
        if (num % i == 0) {  
            return false;  
        }  
    }  
    return true;  
}  
}
```

OUTPUT:



```
D:\>javac SumIndexPrime.java  
  
D:\>java SumIndexPrime  
10  
10  
20  
30  
40  
50  
60  
70  
80  
90  
100  
340
```