Started on	Thursday, 19 October 2023, 12:17 PM
State	Finished
Completed on	Thursday, 19 October 2023, 12:23 PM
Time taken	5 mins 43 secs
Marks	3.00/3.00
Grade	**15.00** out of 15.00 (**100**%)
Name	BALAJI S CSD

Question **1**

Correct

Mark 1.00 out of 1.00

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`

Answer: (penalty regime: 0 %)

```java
import java.util.Scanner;



public class SumNonPrimeIndexes {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);



        int n = scanner.nextInt();



        int[] array = new int[n];



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

            array[i] = scanner.nextInt();

        }



        int sum = calculateSumOfNonPrimeIndexes(array);

        System.out.println( sum);



    }



    public static int calculateSumOfNonPrimeIndexes(int[] arr) {

        int sum = 0;



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

            if (!isPrime(i)) {

                sum += arr[i];

            }

        }



        return sum;

    }



    public static boolean isPrime(int n) {

        if (n <= 1) {

            return false;

        }



        if (n == 2) {

            return true;

        }



        if (n % 2 == 0) {

            return false;

        }



        for (int i = 3; i * i <= n; i += 2) {

            if (n % i == 0) {

                return false;

            }

        }



        return true;

    }

}
```

	Input	Expected	Got	
✓	10 10 20 30 40 50 60 70 80 90 100	340	340	✓

	Input	Expected	Got	
✓	6 -1 -2 -3 3 4 -7	1	1	✓
✓	2 -4 -2	-6	-6	✓

Passed all tests! ✓

Correct
Marks for this submission: 1.00/1.00.

Question **2**

Correct

Mark 1.00 out of 1.00

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 staring 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

Answer: (penalty regime: 0 %)

```java
import java.util.Scanner;



public class FindFirstRepeatedNumber {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);



        int n = scanner.nextInt();



        int[] array = new int[n];





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

            array[i] = scanner.nextInt();

        }



        int result = findFirstRepeatedNumber(array);

        System.out.println( result);



    }



    public static int findFirstRepeatedNumber(int[] arr) {

        int firstRepeated = 0;



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

            if (arr[i] <= 0) {

                continue; // Ignore negative numbers and zeros

            }



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

                if (arr[i] == arr[j]) {

                    firstRepeated = arr[i];

                    return firstRepeated;

                }

            }

        }



        // If no number is repeated, return the first element of the array

        if (arr.length > 0) {

            return arr[0];

        }



        // If all elements are negative or 0, return 0

        return 0;

    }

}

```

	Input	Expected	Got	
✓	6 1 2 4 1 2 8	1	1	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 1.00

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

Answer: (penalty regime: 0 %)

```java
import java.util.Scanner;


public class SumPrimesExceptLargest {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);



        int n = scanner.nextInt();



        int[] array = new int[n];



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

            array[i] = scanner.nextInt();

        }



        int result = sumPrimesExceptLargest(array);

        System.out.println( result);

    }


    public static int sumPrimesExceptLargest(int[] arr) {

        int sum = 0;

        int largestPrime = 0;


        for (int num : arr) {

            if (isPrime(num)) {

                if (num > largestPrime) {

                    sum += largestPrime;

                    largestPrime = num;

                } else {

                    sum += num;

                }

            }

        }


        if (sum == 0) {

            // If there are no prime numbers, sum all numbers except the largest

            int largest = arr[0];

            for (int num : arr) {

                if (num > largest) {

                    largest = num;

                }

            }

            return sumAllExceptLargest(arr, largest);

        }


        return sum;

    }


    public static boolean isPrime(int n) {

        if (n <= 1) {

            return false;

        }


        for (int i = 2; i <= Math.sqrt(n); i++) {

            if (n % i == 0) {

                return false;

            }
        }


        return true;
    }


    public static int sumAllExceptLargest(int[] arr, int largest) {
        int sum = 0;


        for (int num : arr) {
            if (num != largest) {
                sum += num;
            }
        }


        return sum;
    }
}
```

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

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

◄ WEEK_10_MCQ

Jump to...

WEEK_11_MCQ ►