Started on	Thursday, 7 September 2023, 12:06 PM
State	Finished
Completed on	Thursday, 19 October 2023, 10:48 AM
Time taken	41 days 22 hours
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

Akshay works for the Secret Intelligent Services. His role involves decoding cryptic information.

Recently he has received 3 sets of numbers (3 arrays of numbers). The number of elements in each set (in each array) is the same. These numbers have to be decoded to find the room number of a person staying in Hotel Double Cross. The below example demonstrates the steps to be performed while decoding the given numbers.

Example:

Let us assume that the 3 set of numbers are as below:

input1: 4

input2: {1, 2, 3, 4}

input3: {2, 3, 4, 5}

input4: {1, 3, 5, 7}

Step 1:

Generate a new set of elements by adding numbers present at the same index in the three arrays. So, we get, {4, 8, 12, 16}.

Step 2:

The array generates in Step 1 represents the position of the elements in the three arrays i.e.,

The first number 4 represents the number in 4^{th} position in the FIRST array.

The second number 8 represents the number in 8^{th} position in the SECOND array.

The third number 12 represents the number in 12^{th} position in the THIRD array.

The fourth number 16 represents the number in 16^{th} position in the FIRST array.

In Step 2, our task is to pick the numbers from these specified positions from the respective arrays, i.e.,

Number at 4^{th} position in the FIRST array is 4.

Number at 8^{th} position in the SECOND array is 5 (Note that the array contains only 4 elements, so the 8^{th} position can be found by counting the positions in cyclic manner i.e., after reaching last element of the array, continue counting from the first element of the array. This way the number in the 8^{th} position in the second array is 5).

Number at 12^{th} position in the THIRD array is 7 (apply same counting logic as above).

Number at 16^{th} position in the FIRST array is 4 (apply same counting logic as above).

So, we get the positional numbers as {4, 5, 7, 4}.

Step 3:

Add ALL the positional numbers generated in Step 2 to get the FINAL result which represents the room number in Hotel Double Cross, i.e.,

Room number = 4 + 5 + 7 + 4 = 20

Note:

1) There will always be THREE input arrays.
2) All 3 arrays will have the same number of elements.
3) The number of array elements is specified by input4.
4) The array elements will always be positive numbers greater than 0.

Example 2:

Let us now assume the 3 input arrays are as given below:

input1 = 7 (the number of elements in each of the input arrays)

input2 = {10, 33, 5, 40, 120, 98, 1}

input3 = {121, 78, 21, 32, 91, 340, 72}

input4 = {65, 320, 72, 84, 32, 843, 40}

Step 1:

The new set of elements by adding number present at the same index in the three arrays will be {196, 431, 98, 156, 243, 1281, 113}.

Step 2:

Picking up numbers from input1, input2 and input 3 based on the output of Step 1 we get:

1 (number present at position 196 in input1)

32 (number present at position 431 in input2)

40 (number present at position 98 in input3)

33 (number present at position 156 in input1)

91 (number present at position 243 in input2)

40 (number present at position 1281 in input3)

10 (number present at position 113 in input1)

Step 3:

Sum of these numbers gives the Room Number: 247

For example:

Input	Result
4 1 2 3 4 2 3 4 5 1 3 5 7	20
7 10 33 5 40 120 98 1 121 78 21 32 91 340 72 65 320 72 84 32 843 40	247

Answer: (penalty regime: 0 %)

Reset answer

```
import java.util.*;

public class HotelDoubleCrossDecoder {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);


        int numElements = scanner.nextInt();


        int[] input1 = new int[numElements];
        int[] input2 = new int[numElements];
        int[] input3 = new int[numElements];

        for (int i = 0; i < numElements; i++) {
            input1[i] = scanner.nextInt();
        }

        for (int i = 0; i < numElements; i++) {
            input2[i] = scanner.nextInt();
        }

        for (int i = 0; i < numElements; i++) {
            input3[i] = scanner.nextInt();
        }


        int[] resultArray = new int[numElements];
        for (int i = 0; i < numElements; i++) {
            resultArray[i] = input1[i] + input2[i] + input3[i];
        }


        int[] positionalNumbers = new int[numElements];
        for (int i = 0; i < numElements; i++) {
            int position = resultArray[i] % numElements;
            if (position < 0) {

                position += numElements;
            }
            if (position == 0) {
                position = numElements;
            }
            int arrayIndex = position - 1;
```

```
            int index = i % 3;

            if (index == 0) {
                positionalNumbers[i] = input1[arrayIndex];
            } else if (index == 1) {
                positionalNumbers[i] = input2[arrayIndex];
            } else if (index == 2) {
                positionalNumbers[i] = input3[arrayIndex];
            }
        }


        int roomNumber = 0;
        for (int num : positionalNumbers) {
            roomNumber += num;
        }


        System.out.println(roomNumber);

        scanner.close();
    }
}
```

	Input	Expected	Got	
✔	4 1 2 3 4 2 3 4 5 1 3 5 7	20	20	✔
✔	7 10 33 5 40 120 98 1 121 78 21 32 91 340 72 65 320 72 84 32 843 40	247	247	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **2**

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 smallest 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 smallest 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 241, which is the sum of all prime numbers in this array except the smallest prime number 29.

Explanation:

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

The smallest prime number in this array is 29.

So, let us leave out 29 and add all the other prime numbers to get the output.

Therefore, output = 41 + 43 + 31 + 59 + 67 = 241.

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 lowest number.

For example, if input1 = 4 representing the number of elements in the array and input2 = {10, 20, 30, 40} , then the expected output = 20 + 30 + 40 = 90.

For example:

Input	Result
11 10 41 18 50 43 31 29 25 59 96 67	241

Answer: (penalty regime: 0 %)

Reset answer

```
import java.util.*;
class Sample
{
    int n;
    int arr[];
    Sample(int n,int arr[])
    {
        this.n=n;
        this.arr=arr;
    }
    int isPrime(int x)
    {
        int c=0;
        for(int i=1;i<=x;i++)
        {
            if(x%i==0)
            ++c;
        }
        if(c==2)
        return 1;
        else
        return 0;
    }
    public int sumOfPrimeExceptSmall(int arr[], int n)
    {
        //complete this method to solve the goven problem statement
        int sum=0,j=0;
        int ar[]=new int[n];
        for(int i=0;i<n;i++)
        {
            if(isPrime(arr[i])==1)
            {
                ar[j++]=arr[i];
```

```
                sum+=arr[i];
            }
        }

        int small=ar[0];
        for(int i=1;i<ar.length;i++)
        {
            if(ar[i]<small&&ar[i]!=0)
            small=ar[i];
        }
        sum=sum-small;
    return sum;
    }
}
public class SampleDemo
{
    public static void main(String args[])
    {
        //read input and call uncommonElement method
        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();
        Sample obj=new Sample(n,arr);
        int res=obj.sumOfPrimeExceptSmall(arr,n);
        System.out.println(res);
    }
}
```

	Input	Expected	Got	
✔	11 10 41 18 50 43 31 29 25 59 96 67	241	241	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 1.00

Given a number, you are expected to find its single-digit "Reduced Subtracted Form (RSF)".

The "Reduced Subtracted Form (RSF)" of a number can be found by concatenating the difference between its adjacent digits.

To find the single-digit "Reduced Subtracted Form (RSF)", we need to continue this process till the resultant RSF is not a single digit.

For eg. If the number is 6928, its RSF can be found by concatenating the difference between (6 and 9), (9 and 2) and (2 and 8) as shown below:

difference between 6 and 9 is 3

difference between 9 and 2 is 7

difference between 2 and 8 is 6

So, the "Reduced Subtracted Form (RSF)" of 6928 = 376.

The resultant RSF (376) is not a single-digit, so we must continue finding its "Reduced Subtracted Form (RSF)".

difference between 3 and 7 is 4

difference between 7 and 6 is 1

So, the "Reduced Subtracted Form (RSF)" of 376 = 41.

The resultant RSF (41) is not a single-digit, so we must continue finding its "Reduced Subtracted Form (RSF)".

difference between 4 and 1 is 3

The resultant number (3) is a single-digit, so we have reached the "single-digit Reduced Subtracted Form"

Therefore, the single-digit RSF of 6928 = 3.

Let's see another example:

If input1 = 5271

Expected Output = 1

Explanation:

RSF of 5271 = (5 – 2) (2 – 7) (7 – 1) = 356

RSF of 356 = (3 – 5) (5 – 6) = 21

RSF of 21 = (2 – 1) = 1

Note1: input1 will always be >= 10.

Note2: Note that while concatenating the differences, we are expected to use the absolute values (non-negative).

For example:

Input	Result
6928	3

Answer: (penalty regime: 0 %)

Reset answer

```
import java.util.*;
class Sample
{
    int n;
    Sample(int n)
    {
        this.n=n;
    }
    int length(int a)
    {
        int c=0;
        while(a!=0)
        {
            a=a/10;
            ++c;
        }
        return c;
    }
```

```
    int reducedform(int b)
    {
        int s=0;
        int l=length(b);
        int arr[]=new int[l];
        for(int i=l-1;i>=0;i--)
        {
            arr[i]=b%10;
            b=b/10;
        }
        for(int i=0;i<l-1;i++)
        {
            s=s*10+(Math.abs(arr[i]-arr[i+1]));
        }
        return s;
    }
    public int rsf(int n)
    {
        int res=reducedform(n);
        while(res>9)
        {
            res=reducedform(res);
        }
        return res;
    }
}
public class SampleDemo
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        Sample obj=new Sample(n);
        System.out.print(obj.rsf(n));
    }
}
```

	Input	Expected	Got	
✔	6928	3	3	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ WEEK_04_MCQ

Jump to...

Classes - Offine Excercise ►