

CS23333-Object Oriented Programming Using Java-2023

Dashboard / My courses / CS23333-OOPJ-2023 / Lab-03-Arrays / Lab-03-Logic Building

Quiz navigation




Show one page at a time
Finish review

Status	Finished
Started	Sunday, 6 October 2024, 12:38 AM
Completed	Sunday, 6 October 2024, 12:40 AM
Duration	2 mins 29 secs

Question **1**

Correct

Marked out of 5.00

 Flag question

Given an integer array as input, perform the following operations on the array, in the below specified sequence.

- Find the maximum number in the array.
- Subtract the maximum number from each element of the array.
- Multiply the maximum number (found in step 1) to each element of the resultant array.

After the operations are done, return the resultant array.

Example 1:

input1 = 4 (represents the number of elements in the input1 array)

input2 = {1, 5, 6, 9}

Expected Output = {-72, -36, 27, 0}

Explanation:

Step 1: The maximum number in the given array is 9.

Step 2: Subtracting the maximum number 9 from each element of the array:

{(1 - 9), (5 - 9), (6 - 9), (9 - 9)} = {-8, -4, -3, 0}

Step 3: Multiplying the maximum number 9 to each of the resultant array:

{(-8 x 9), (-4 x 9), (3 x 9), (0 x 9)} = {-72, -36, -27, 0}

So, the expected output is the resultant array {-72, -36, -27, 0}.

Example 2:

input1 = 5 (represents the number of elements in the input1 array)

input2 = {10, 87, 63, 42, 2}

Expected Output = {-6699, 0, -2088, -3915, -7395}

Explanation:

Step 1: The maximum number in the given array is 87.

Step 2: Subtracting the maximum number 87 from each element of the array:

{(10 - 87), (87 - 87), (63 - 87), (42 - 87), (2 - 87)} = {-77, 0, -24, -45, -85}

Step 3: Multiplying the maximum number 87 to each of the resultant array:

{(-77 x 87), (0 x 87), (-24 x 87), (-45 x 87), (-85 x 87)} = {-6699, 0, -2088, -3915, -7395}

So, the expected output is the resultant array {-6699, 0, -2088, -3915, -7395}.

Example 3:

input1 = 2 (represents the number of elements in the input1 array)

input2 = {-9, 9}

Expected Output = {-162, 0}

Explanation:

Step 1: The maximum number in the given array is 9.

Step 2: Subtracting the maximum number 9 from each element of the array:

{(-9 - 9), (9 - 9)} = {-18, 0}

Step 3: Multiplying the maximum number 9 to each of the resultant array:

{(-18 x 9), (0 x 9)} = {-162, 0}

So, the expected output is the resultant array {-162, 0}.

Note: The input array will contain not more than 100 elements

For example:

Input	Result
4 1 5 6 9	-72 -36 -27 0
5 10 87 63 42 2	-6699 0 -2088 -3915 -7395
2 -9 9	-162 0

Answer: (penalty regime: 0 %)

```
1 import java.util.Scanner;
2 public class res{
3     public static int[] pa(int[] arr){
4         int maxs=Integer.MIN_VALUE;
5         for (int num:arr){
6             if(num>maxs){
7                 maxs=num;
8             }
9         }
10        for(int i=0;i<arr.length;i++){
11            arr[i]=(arr[i]-maxs)*maxs;
12        }
13        return arr;
14    }
```

```

15 ✓ public static void main(String[] args){
16     Scanner scanner =new Scanner (System.in);
17     int n=scanner.nextInt();
18     int[]jarr=new int[n];
19     for(int i=0;i<n;i++){
20         arr[i]=scanner.nextInt();
21     }
22     int[]res=pa(arr);
23     for(int i=0;i<n;i++){
24         System.out.print(res[i]+" ");
25     }
26     scanner.close();
27 }
28 }
29 }

```

	Input	Expected	Got
	4 1 5 6 9	-72 -36 -27 0	-72 -36 -27 0
	5 10 87 63 42 2	-6699 0 -2088 -3915 -7395	-6699 0 -2088 -3915 -7395
	2 -9 9	-162 0	-162 0

Passed all tests!

Question **2**

Correct

Marked out of 5.00

[Flag question](#)

You are provided with a set of numbers (array of numbers).

You have to generate the sum of specific numbers based on its position in the array set provided to you.

This is explained below:

Example 1:

Let us assume the encoded set of numbers given to you is:

input1:5 and input2: {1, 51, 436, 7860, 41236}

Step 1:

Starting from the 0th index of the array pick up digits as per below:

0th index – pick up the units value of the number (in this case is 1).

1st index - pick up the tens value of the number (in this case it is 5).

2nd index - pick up the hundreds value of the number (in this case it is 4).

3rd index - pick up the thousands value of the number (in this case it is 7).

4th index - pick up the ten thousands value of the number (in this case it is 4).

(Continue this for all the elements of the input array).

The array generated from Step 1 will then be – {1, 5, 4, 7, 4}.

Step 2:

Square each number present in the array generated in Step 1.

{1, 25, 16, 49, 16}

Step 3:

Calculate the sum of all elements of the array generated in Step 2 to get the final result. The result will be = 107.

Note:

- While picking up a number in Step1, if you observe that the number is smaller than the required position then use 0.
- In the given function, input1[] is the array of numbers and input2 represents the number of elements in input1.

Example 2:

input1: 5 and input1: {1, 5, 423, 310, 61540}

Step 1:

Generating the new array based on position, we get the below array:

{1, 0, 4, 0, 6}

In this case, the value in input1 at index 1 and 3 is less than the value required to be picked up based on position, so we use a 0.

Step 2:

{1, 0, 16, 0, 36}

Step 3:

The final result = 53.

For example:

Input	Result
5 1 51 436 7860 41236	107
5 1 5 423 310 61540	53

Answer: (penalty regime: 0 %)

```

1 ✓ import java.util.Scanner;
2 public class digit{
3     public static void main(String[] args){
4         Scanner scanner =new Scanner(System.in);
5         int size =scanner.nextInt();
6         int[]inpar=new int[size];
7         for(int i=0;i<size;i++){
8             inpar[i]=scanner.nextInt();
9         }
10        int[]dig=new int[size];
11        for(int i=0;i<size;i++){
12            int num=inpar[i];

```

```

13     if(i==0){
14         dig[i]=num%10;
15     }
16     else if (i==1){
17         dig[i]=(num/10)%10;
18     }
19     else if(i==2){
20         dig[i]=(num/100)%10;
21     }
22     else if(i==3){
23         dig[i]=(num/1000)%10;
24     }
25     }
26     else if(i==4){
27         dig[i]=(num/10000)%10;
28     }
29     else{
30         dig[i]=0;
31     }
32 }
33 int fin=0;
34 for(int digi:dig){
35     fin+=digi*digi;
36 }
37 System.out.print(fin);
38 }
39 }
40 }

```

Input	Expected	Got
5 1 51 436 7860 41236	107	107
5 1 5 423 310 61540	53	53

Passed all tests!

Question 3

Correct

Marked out of 5.00

[Flag question](#)

Given an array of numbers, you are expected to return the sum of the longest sequence of POSITIVE numbers in the array.

If there are NO positive numbers in the array, you are expected to return -1.

In this question's scope, the number 0 should be considered as positive.

Note: If there are more than one group of elements in the array having the longest sequence of POSITIVE numbers, you are expected to return the total sum of all those POSITIVE numbers (see example 3 below).

input1 represents the number of elements in the array.

input2 represents the array of integers.

Example 1:

input1 = 16

input2 = {-12, -16, 12, 18, 18, 14, -4, -12, -13, 32, 34, -5, 66, 78, 78, -79}

Expected output = 62

Explanation:

The input array contains four sequences of POSITIVE numbers, i.e. "12, 18, 18, 14", "12", "32, 34", and "66, 78, 78". The first sequence "12, 18, 18, 14" is the longest of the four as it contains 4 elements. Therefore, the expected output = sum of the longest sequence of POSITIVE numbers = 12 + 18 + 18 + 14 = 63.

Example 2:

input1 = 11

input2 = {-22, -24, 16, -1, -17, -19, -37, -25, -19, -93, -61}

Expected output = -1

Explanation:

There are NO positive numbers in the input array. Therefore, the expected output for such cases = -1.

Example 3:

input1 = 16

input2 = {-58, 32, 26, 92, -10, -4, 12, 0, 12, -2, 4, 32, -9, -7, 78, -79}

Expected output = 174

Explanation:

The input array contains four sequences of POSITIVE numbers, i.e. "32, 26, 92", "12, 0, 12", "4, 32", and "78". The first and second sequences "32, 26, 92" and "12, 0, 12" are the longest of the four as they contain 4 elements each. Therefore, the expected output = sum of the longest sequence of POSITIVE numbers = (32 + 26 + 92) + (12 + 0 + 12) = 174.

For example:

Input	Result
16 -12 -16 12 18 18 14 -4 -12 -13 32 34 -5 66 78 78 -79	62
11 -22 -24 -16 -1 -17 -19 -37 -25 -19 -93 -61	-1
16 -58 32 26 92 -10 -4 12 0 12 -2 4 32 -9 -7 78 -79	174

Answer: (penalty regime: 0 %)

```

1 import java.util.Scanner;
2 public class longdig{
3     public static void main(String[] args){
4         Scanner sc=new Scanner(System.in);
5         int n=sc.nextInt();

```

```

6      int c = 1,v,seqtemp = 0,seq = 0,countmax = 0;
7      int count = 0;
8      while(c <= n){
9          v = sc.nextInt();
10         if(v >= 0){
11             countmax= countmax + v;
12             seqtemp++;
13         }
14         else{
15             seqtemp = 0;
16             countmax = 0;
17         }
18         if(seqtemp > seq ){
19             seq = seqtemp;
20             count = countmax;
21         }
22         else if (seq == seqtemp){
23             count = count + countmax;
24         }
25         c++;
26     }
27     if (count == 0)
28         System.out.print(-1);
29     else
30         System.out.print(count);
31 }
32 }
33

```

Input	Expected	Got	
16 -12 -16 12 18 18 14 -4 -12 -13 32 34 -5 66 78 78 -79	62	62	
11 -22 -24 -16 -1 -17 -19 -37 -25 -19 -93 -61	-1	-1	
16 -58 32 26 92 -10 -4 12 0 12 -2 4 32 -9 -7 78 -79	174	174	

Passed all tests!

[Finish review](#)

[◀ Lab-03-MCQ](#)

Jump to...

[Simple Encoded Array ▶](#)