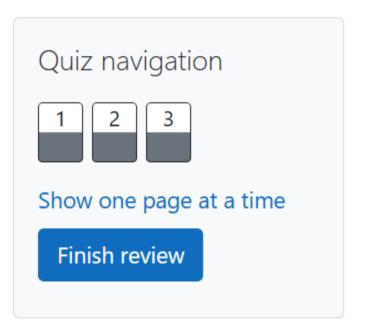
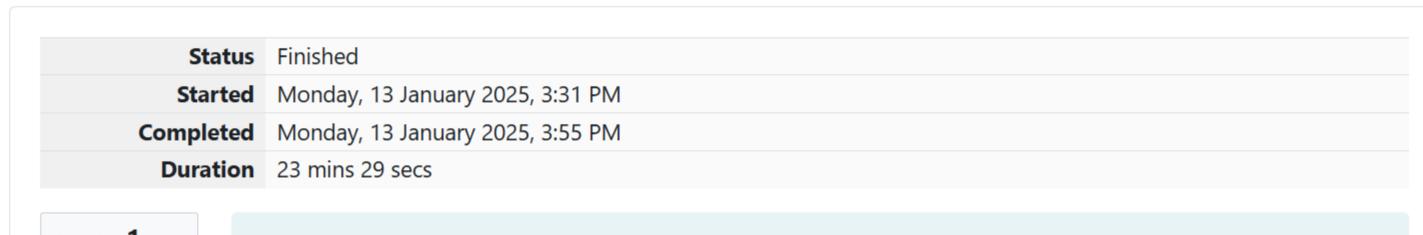
GE23131-Programming Using C-2024





Question **1**

Correct

▼ Flag question

Given an array of numbers, find the index of the smallest array element (the pivot), for which the sums of all elements to the left and to the right are equal. The array may not be reordered.

Example

arr=[1,2,3,4,6]

- the sum of the first three elements, 1+2+3=6. The value of the last element is 6.
- Using zero based indexing, arr[3]=4 is the pivot between the two subarrays.
- The index of the pivot is 3.

Function Description

Complete the function balancedSum in the editor below.

balancedSum has the following parameter(s):

int arr[n]: an array of integers

Returns:

int: an integer representing the index of the pivot

Constraints

- $3 \le n \le 10^5$
- · $1 \le arr[i] \le 2 \times 10^4$, where $0 \le i < n$
- · It is guaranteed that a solution always exists.

Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer n, the size of the array arr.

Each of the next n lines contains an integer, arr[i], where $0 \le i < n$.

Sample Case 0 Sample Input 0 STDIN Function Parameters \rightarrow arr[] size n = 4 \rightarrow arr = [1, 2, 3, 3] 2 3 3 Sample Output 0 2 Explanation 0 The sum of the first two elements, 1+2=3. The value of the last element is 3. Using zero based indexing, arr[2]=3 is the pivot between the two subarrays. The index of the pivot is 2.

Sample Case 1 Sample Input 1 STDIN Function Parameters \rightarrow arr[] size n = 3 \rightarrow arr = [1, 2, 1] 2 Sample Output 1 Explanation 1 The first and last elements are equal to 1. Using zero based indexing, arr[1]=2 is the pivot between the two subarrays. The index of the pivot is 1.

Answer: (penalty regime: 0 %)

Reset answer

```
* Complete the 'balancedSum' function below.
     * The function is expected to return an INTEGER.
     * The function accepts INTEGER_ARRAY arr as parameter.
 6
    int balancedSum(int arr_count, int* arr)
9 ₹ {
        int left=0,right=0;
10
        for(int i=0;i<arr_count;i++)</pre>
11
12 ₹
            right+=arr[i];
13
14
        for(int i=0;i<arr_count;i++)</pre>
15
16 ₹
17
            if(left==right-arr[i])
18
            return i;
19
20
        left+=arr[i];
21
        right-=arr[i];
22
23
24
        return 1;
25
26
27
28
29
```

```
17
           if(left==right-arr[i])
18
           return i;
19
20
        left+=arr[i];
21
        right-=arr[i];
22
23
        return 1;
24
25
26
27
28
29
```

	Test	Expected	Got	
~	<pre>int arr[] = {1,2,3,3}; printf("%d", balancedSum(4, arr))</pre>	2	2	~

Passed all tests! ✓

Question **2**

Correct

Flag question

Calculate the sum of an array of integers.

Example

numbers = [3, 13, 4, 11, 9]

The sum is 3 + 13 + 4 + 11 + 9 = 40.

Function Description

Complete the function arraySum in the editor below.

arraySum has the following parameter(s):

int numbers[n]: an array of integers

Returns

int: integer sum of the numbers array

Constraints

Constraints

$$1 \le n \le 10^4$$

 $1 \le numbers[i] \le 10^4$

Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer n, the size of the array numbers.

Each of the next n lines contains an integer numbers[i] where $0 \le i < n$.

Sample Case 0

Sample Input 0

STDIN Function

- $5 \rightarrow numbers[] size n = 5$
- $1 \rightarrow numbers = [1, 2, 3, 4, 5]$

2

```
\rightarrow numbers[] size n = 5
     \rightarrow numbers = [1, 2, 3, 4, 5]
2
3
4
5
Sample Output 0
15
Explanation 0
1 + 2 + 3 + 4 + 5 = 15.
Sample Case 1
Sample Input 1
STDIN Function
     \rightarrow numbers[] size n = 2
      \rightarrow numbers = [12, 12]
```

Sample Case 1

Sample Input 1

STDIN Function

 $2 \rightarrow numbers[] size n = 2$

 $12 \rightarrow numbers = [12, 12]$

12

Sample Output 1

24

Explanation 1

12 + 12 = 24.

Answer: (penalty regime: 0 %)

Reset answer

```
Reset answer
```

```
* Complete the 'arraySum' function below.
     * The function is expected to return an INTEGER.
     * The function accepts INTEGER_ARRAY numbers as parameter.
   int arraySum(int numbers_count, int *numbers)
9 ₹ {
        int s=0,i=0;
10
        while(i<numbers_count)</pre>
11
12 ₹
            s+=numbers[i];
13
            i++;
14
15
16
        return s;
17
18
19
```

	Test	Expected	Got	
~	<pre>int arr[] = {1,2,3,4,5}; printf("%d", arraySum(5, arr))</pre>	15	15	~

Passed all tests! <

Question ${\bf 3}$

Correct

Flag question

Answer: (penalty regime: 0 %)

Reset answer

```
int minDiff(int arr_count, int* arr)
 9 ₹ {
         if(arr_count<=1)</pre>
10
11
        return 0;
         void q_sort(int arr[],int low,int high)
12
13 🔻
             if(low<high)</pre>
14
15 v
                 int pivot=arr[high];
16
                 int i=low-1;
17
                 for(int j=low;j<high;j++)</pre>
18
19
                      if(arr[j]<pivot)</pre>
20
21
22
                          i++;
                          int temp=arr[i];
23
                          arr[i]=arr[j];
24
                          arr[j]=temp;
25
26
27
                 int temp=arr[i+1];
28
                 arr[i+1]=arr[high];
29
30
                 arr[high]=temp;
31
                 q_sort(arr,low,i);
                 q_sort(arr,i+2,high);
32
33
34
35
         q_sort(arr,0,arr_count-1);
36
         int s=0;
         for(int i=1;i<arr_count;i++)</pre>
37
38 🔻
39
             s+=arr[i]-arr[i-1];
40
41
         return s;
42
```

```
int temp=arr[1+1];
                arr[i+1]=arr[high];
                arr[high]=temp;
                q_sort(arr,low,i);
                q_sort(arr,i+2,high);
32
33
34
        q_sort(arr,0,arr_count-1);
        int s=0;
36
        for(int i=1;i<arr_count;i++)</pre>
37
38 ₹
            s+=arr[i]-arr[i-1];
39
40
        return s;
41
42
43
```

	Test	Expected	Got	
~	<pre>int arr[] = {5, 1, 3, 7, 3}; printf("%d", minDiff(5, arr))</pre>	6	6	~

Passed all tests! ✓