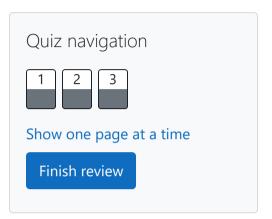
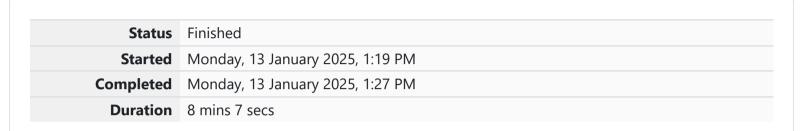
GE23131-Programming Using C-2024





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.

Returns:

int: an integer representing the index of the pivot

Constraints

- $\cdot \qquad 3 \le n \le 10^5$
- \cdot 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

$$4 \rightarrow arr[] size n = 4$$

 $1 \rightarrow arr = [1, 2, 3, 3]$

Sample Output 0 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] Sample Output 1

3

2

2

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

```
1 | /*
     * Complete the 'balancedSum' function below.
 2
 3
     * The function is expected to return an INTEGER.
     * The function accepts INTEGER ARRAY arr as parameter.
 5
 6
    int balancedSum(int arr_count, int* arr)
 9 1
        int totalsum=0,leftsum=0;
10
        for(int i=0;i<arr_count;i++){</pre>
11 🔻
             totalsum+=arr[i];
12
13
        for (int i=0;i<arr_count;i++){</pre>
14 •
             totalsum-=arr[i];
15
16
             if(leftsum==totalsum){
                 return i;
17
18
             leftsum+=arr[i];
19
20
        return 1;
21
22
23
```

Question **2**

▼ Flag question

Correct

	<pre>int arr[] = {1,2,3,3}; printf("%d", balancedSum(4, arr))</pre> 2 2				
Passed all tests! ✓					
	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				

```
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
     \rightarrow numbers[] size n = 5
     \rightarrow numbers = [1, 2, 3, 4, 5]
2
5
Sample Output 0
15
```

```
1 + 2 + 3 + 4 + 5 = 15.
Sample Case 1
Sample Input 1
STDIN Function
   → numbers[] size n = 2
12 \rightarrow \text{numbers} = [12, 12]
12
Sample Output 1
24
Explanation 1
12 + 12 = 24.
Answer: (penalty regime: 0 %)
  Reset answer
```

* Complete the 'arraySum' function below.

* The function is expected to return an INTEGER.

* The function accepts INTEGER ARRAY numbers as parameter.

```
9 * {
    int sum=0;
    for(int i=0;i<numbers_count;i++){
        sum+=numbers[i];
    }
    return sum;
}
```

	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

Given an array of n integers, rearrange them so that the sum of the absolute differences of all adjacent elements is minimized. Then, compute the sum of those absolute differences. Example n = 5 arr = [1, 3, 3, 2, 4] If the list is rearranged as arr' = [1, 2, 3, 3, 4], the absolute differences are |1 - 2| = 1, |2 - 3| = 1, |3 - 3| = 0, |3 - 4| = 1. The sum of those differences is 1 + 1 + 0 + 1 = 3. Function Description Complete the function minDiff in the editor below. minDiff has the following parameter: arr: an integer array Returns: int: the sum of the absolute differences of adjacent elements Constraints $2 \le n \le 105$ $0 \le arr[i] \le 109$, where $0 \le i < n$ Input Format For Custom Testing The first line of input contains an integer, n, the size of arr. Each of the

5 5 \rightarrow arr[] = [5, 1, 3, 7, 3] 1 3 7 3 Sample Output 6 Explanation n = 5 arr = [5, 1, 3, 7, 3] If arr is rearranged as arr' = [1, 3, 3, 5, 7], the differences are minimized. The final answer is |1 - 3| + |3 - 3| + |3 - 5| + |5 - 7| = 6. Sample Case 1 Sample Input For Custom Testing STDIN Function ----- 2 \rightarrow arr[] size n = 2 3 \rightarrow arr[] = [3, 2] 2 Sample Output 1 Explanation n = 2 arr = [3, 2] There is no need to rearrange because there are only two elements. The final answer is |3 - 2| = 1.

Answer: (penalty regime: 0 %)

Reset answer

```
1 🔻
     * Complete the 'minDiff' function below.
 2
 3
     * The function is expected to return an INTEGER.
 4
     * The function accepts INTEGER ARRAY arr as parameter.
 6
    int compare(const void* a,const void*b){
        return(*(int*)a-*(int*)b);
 8
 9
    int minDiff(int arr count, int* arr)
10
11 🔻
        qsort(arr,arr_count,sizeof(int),compare);
12
        int min sum=0;
13
        for(int i=1;i<arr count;i++)</pre>
14
        min sum+=abs(arr[i]-arr[i-1]);
15
        return min sum;
16
17
18
```

Test Expected Got

