

# GE23131-Programming Using C-2024

<b>Status</b>	Finished
<b>Started</b>	Tuesday, 14 January 2025, 11:38 AM
<b>Completed</b>	Tuesday, 14 January 2025, 11:56 AM
<b>Duration</b>	18 mins 4 secs

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,  $\text{arr}[3]=4$  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

- $3 \leq n \leq 10^5$
- $1 \leq \text{arr}[i] \leq 2 \times 10^4$ , where  $0 \leq i < n$

- 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  $\text{arr}$ .

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

## Sample Case 0

### Sample Input 0

## STDIN Function Parameters

-----

4 →  $\text{arr}[]$  size  $n = 4$

1 →  $\text{arr} = [1, 2, 3, 3]$

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,  $\text{arr}[2]=3$  is the pivot between the two subarrays.
- The index of the pivot is 2.

## Sample Case 1

### Sample Input 1

### STDIN    Function Parameters

-----

3 → arr[] size n = 3

1 → arr = [1, 2, 1]

2

1

## 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

```
1  /*
2  * Complete the 'balancedSum'
3  *
4  * The function is expected
5  * The function accepts INTE
6  */
7
8 int balancedSum(int arr_coun
9 {
10     int left = 0, right = 0
11     for(int i=0; i<arr_coun
12         right += arr[i];
13     }
14     for(int i=0;i<arr_count
15         if(left==(right-arr
16             return i;
```

**Answer:** (penalty regime: 0 %)

Reset answer

```
1
2 complete the 'balancedSum' function
3
4 The function is expected to return an integer.
5 The function accepts INTEGER_ARA
6
7
8 balancedSum(int arr_count, int arr[])
9
10    int left = 0, right = 0;
11    for(int i=0; i<arr_count;i++)
12        right += arr[i];
13    }
14    for(int i=0;i<arr_count;i++)
15        if(left==(right-arr[i]))
16            return i;
17        left += arr[i];
18        right -= arr[i];
19    }
20    return 1;
21
22
```

Test	
✓	int arr[] = {1,2,3,3}; printf("%d", balancedSum(4, arr));



```
3
4     to return an INTEGER.
5     EGER_ARRAY arr as parameter.
6
7
8     nt, int* arr)
9
10    0;
11    nt; i++) {
12
13
14    t; i++) {
15    r[i]))
```

	<b>Expected</b>	<b>Got</b>	
,3,3}; ncedSum(4, arr))	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

`int: integer sum of the numbers array`

## Constraints

$1 \leq n \leq 10^4$

$1 \leq \text{numbers}[i] \leq 10^4$

## 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  $0 \leq i < n$ .

## Sample Case 0

### Sample Input 0

### STDIN      Function

-----

5       $\rightarrow$  `numbers[]` size  $n = 5$

1       $\rightarrow$  `numbers = [1, 2, 3, 4, 5]`

2

3

4

5

## Sample Output 0

15

## Explanation 0

$$1 + 2 + 3 + 4 + 5 = 15.$$

## Sample Case 1

## Sample Input 1

STDIN      Function

----- -----

2    →    numbers[] size n = 2

12    →    numbers = [12, 12]

12

## Sample Output 1

## Sample Output 1

24

## Explanation 1

$$12 + 12 = 24.$$

**Answer:** (penalty regime: 0 %)

Reset answer

```
1  /*
2   * Complete the 'arraySum' f
3   *
4   * The function is expected
5   * The function accepts INTE
6   */
7
8 int arraySum(int numbers_cou
9 {
10    int sum=0;
11    for(int i=0;i < numbers_
12      sum += numbers[i];
13    }
14    return sum;
15 }
16 }
```

```
2 sum function below.
3
4 expected to return an INTEGER.
5 as INTEGER_ARRAY numbers as pa
6
7
8 int arraySum(int s_count, int *numbers)
9
10
11 for (int i = 0; i < s_count; i++) {
12     arr[i];
13
14
15
16 }
```

### Test

✓	int arr[] = {1,2,3,4,5}; printf("%d", arraySum(5, arr))
---	--

Passed all tests! ✓

Question 3

Correct

Flag question

### Question 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  $1 + 1 + 0 + 1 = 3$ .

**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**  $2 \leq n \leq 105$   $0 \leq \text{arr}[i] \leq 109$ , where  $0 \leq i < n$

**Input Format** For Custom Testing The first line of input contains an integer,  $n$ , the size of arr. Each of the following  $n$  lines contains an integer that describes  $\text{arr}[i]$  (where  $0 \leq i < n$ ) .

**Sample Case 0 Sample Input For Custom Testing**

STDIN Function ----- 5 → arr[] size  $n = 5$

5 → 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 → arr[] size n = 2  
3 → 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 /*  
2  * Complete the 'minDiff' fu  
3  *  
4  * The function is expected  
5  * The function accepts INTE  
6  */  
7  
8 int minDiff(int arr_count, i  
9 {  
10    for(int i=0; i<arr_count  
11      for(int j=0;j<arr_co  
12        if(arr[j]>arr[j+  
13          int temp = a  
14          arr[j]=arr[j  
15          arr[j+1]=tem  
16        }  
17      }  
18    }  
19    int sum=0;  
20    for(int i=0;i<arr_count-  
21      sum+=abs(arr[i]-arr[
```

```
6
7
8 f(int arr_count, int* arr)
9
10 t i=0; i<arr_count-1;i++){
11 r(int j=0;j<arr_count-i-1;j+-
12 if(arr[j]>arr[j+1]){
13     int temp = arr[j];
14     arr[j]=arr[j+1];
15     arr[j+1]=temp;
16 }
17
18
19 m=0;
20 t i=0;i<arr_count-1;i++){
21 m+=abs(arr[i]-arr[i+1]);
22
23 sum;
24
25
```

Test	
✓	int arr[] = {5, 1, 3, 7, 3}; printf("%d", minDiff(5, arr))

Passed all tests! ✓