

```
4 → arr[] size n = 4
1 → 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

1

Explanation 1

- The first and last elements are equal to 1.
- Using zero based indexing,  $\text{arr}[1]=2$  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' function below.
3   *
4   * The function is expected to return an integer.
5   * The function accepts INTEGER_ARRAY arr and integer n as parameters.
6   */
7
8  int balancedSum(int arr_count, int* arr)
9  {
10     int totalsum=0;
11     for(int i=0;i<arr_count;i++){
12         totalsum+=arr[i];
13     }
14     int leftsum=0;
15     for(int i=0;i<arr_count;i++){
16         int rightsum=totalsum-leftsum-arr[i];
17         if(leftsum==rightsum){
18             return i;
19         }
20         leftsum+=arr[i];
21     }
22     return 1;
23 }
24
```

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

Passed all tests! ✓

### 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

24

### Explanation 1

12 + 12 = 24.

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

Reset answer

```
1  /*
2  * Complete the 'arraySum' function below
3  *
4  * The function is expected to return an
5  * The function accepts INTEGER_ARRAY num
6  */
7
8  int arraySum(int numbers_count, int *numb
9  {
10     int sum=0;
11     for(int i=0;i<numbers_count;i++){
12         sum=sum+numbers[i];
13     }
14     return sum;
15 }
16
```

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

Passed all tests! ✓

### 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  $1 + 1 + 0 + 1 = 3$ . 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  $2 \leq n \leq 105$   $0 \leq arr[i] \leq 109$ , 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 `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  $|1 - 3| + |3 - 3| + |3 - 5| + |5 - 7|$ .

```

1  /*
2   * Complete the 'arraySum' function below
3   *
4   * The function is expected to return an
5   * The function accepts INTEGER_ARRAY num
6   */
7
8  int arraySum(int numbers_count, int *numb
9  {
10     int sum=0;
11     for(int i=0;i<numbers_count;i++){
12         sum=sum+numbers[i];
13     }
14     return sum;
15 }
16

```

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

Passed all tests! ✓

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  $1 + 1 + 0 + 1 = 3$ . 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  $2 \leq n \leq 105$   $0 \leq arr[i] \leq 109$ , 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 `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' function below.
3   *
4   * The function is expected to return an
5   * The function accepts INTEGER_ARRAY arr
6   */
7
8  #include<stdlib.h>
9  int compare(const void *a,const void *b){
10     return(*(int*)a - *(int*)b);
11 }
12 int minDiff(int arr_count, int* arr)
13 {
14     qsort(arr,arr_count,sizeof(int),compa
15     int totaldiff=0;
16     for(int i=1;i<arr_count;i++){
17         totaldiff+=abs(arr[i]-arr[i-1]);
18     }
19     return totaldiff;
20 }
21

```

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

Passed all tests! ✓

Finish review