

**240801240**

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  $0 \leq i < n$ .

---

Sample Input:

STDIN Function Parameters

-----

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.

```

1  /*
2  * Complete the 'balancedSum' function below.
3  *
4  * The function is expected to return an INTEGER.
5  * The function accepts INTEGER_ARRAY arr as parameter.
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	Got	
✓	int arr[] = {1,2,3,3};	2	2	✓

## Question 2: Sum Them All

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:

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

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 Input

5 → `numbers[]` size `n = 5`

1 → `numbers = [1, 2, 3, 4, 5]`

2

3

4

5

Sample Output

15

---

## Explanation

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

```
1  /*
2   * Complete the 'arraySum' function below.
3   *
4   * The function is expected to return an INTEGER.
5   * The function accepts INTEGER_ARRAY numbers as parameter.
6   */
7
8  int arraySum(int numbers_count, int *numbers)
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};	15	15	✓

## Question 3: Minimum Difference Sum

---

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  $0 \leq i < n$

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 Input For Custom Testing

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

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  $|1 - 3| + |3 - 3| + |3 - 5| + |5 - 7| = 6$ .

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

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